You

I'm getting this error when running mySQL in Xampp [Uploaded Image Expired]

Copilot

I'm sorry to hear you're having trouble with MySQL in XAMPP! The error message "MySQL shutdown unexpectedly" can be caused by a few different issues. Here are some steps you can try to resolve it:

1. **Run XAMPP as Administrator**: Sometimes, MySQL needs administrative privileges to run properly. Right-click on the XAMPP icon and select "Run as administrator."

2. **Check for Port Conflicts**: Ensure that no other services are using the same port as MySQL. You can change the port in the my.ini file located in the mysql folder of your XAMPP installation.

3. **Restore Database Backup**: If your database is corrupted, restoring from a backup might help. Make a copy of the data folder in the mysql directory, delete the corrupted data, and replace it with the backup.

4. **Check Error Logs**: Click on the "Logs" button in the XAMPP Control Panel to view detailed error logs. This can give you more clues about what's causing the issue.

5. **Reinstall MySQL**: If all else fails, you might need to reinstall MySQL. Uninstall MySQL from the XAMPP Control Panel, then reinstall it.

If you need more detailed instructions or run into any issues, feel free to ask! I'm here to help.

You

These are the logs:

2024-10-27 17:20:30 0 [Note] Starting MariaDB 10.4.32-MariaDB source revision c4143f909528e3fab0677a28631d10389354c491 as process 15384Cannot find checkpoint record at LSN (1,0x5556)2024-10-27 17:20:30 0 [ERROR] mysqld.exe: Aria recovery failed. Please run aria_chk -r on all Aria tables and delete all aria_log.######## files2024-10-27 17:20:30 0 [ERROR] Plugin 'Aria' registration as a STORAGE ENGINE failed.2024-10-27 17:20:30 0 [Note] InnoDB: Mutexes and rw_locks use Windows interlocked functions2024-10-27 17:20:30 0 [Note] InnoDB: Uses event mutexes2024-10-27 17:20:30 0 [Note] InnoDB: Compressed tables use zlib 1.32024-10-27 17:20:30 0 [Note] InnoDB: Number of pools: 12024-10-27 17:20:30 0 [Note] InnoDB: Using SSE2 crc32 instructions2024-10-27 17:20:30 0 [Note] InnoDB: Initializing buffer pool, total size = 16M, instances = 1, chunk size = 16M2024-10-27 17:20:30 0 [Note] InnoDB: Completed initialization of buffer pool2024-10-27 17:20:30 0 [Note] InnoDB: 128 out of 128 rollback segments are active.2024-10-27 17:20:30 0 [Note] InnoDB: Creating shared tablespace for temporary tables2024-10-27 17:20:30 0 [Note] InnoDB: Setting file 'C:\xampp\mysql\data\ibtmp1' size to 12 MB. Physically writing the file full; Please wait ...2024-10-27 17:20:30 0 [Note] InnoDB: File 'C:\xampp\mysql\data\ibtmp1' size is now 12 MB.2024-10-27 17:20:30 0 [Note] InnoDB: Waiting for purge to start2024-10-27 17:20:31 0 [Note] InnoDB: 10.4.32 started; log sequence number 575610; transaction id 4052024-10-27 17:20:31 0 [Note] InnoDB: Loading buffer pool(s) from C:\xampp\mysql\data\ib_buffer_pool2024-10-27 17:20:31 0 [Note] Plugin 'FEEDBACK' is disabled.2024-10-27 17:20:31 0 [ERROR] Could not open mysql.plugin table. Some plugins may be not loaded2024-10-27 17:20:31 0 [ERROR] Failed to initialize plugins.2024-10-27 17:20:31 0 [ERROR] Aborting2024-10-27 17:20:35 0 [Note] Starting MariaDB 10.4.32-MariaDB source revision c4143f909528e3fab0677a28631d10389354c491 as process 18624Cannot find checkpoint record at LSN (1,0x5556)2024-10-27 17:20:35 0 [ERROR] mysqld.exe: Aria recovery failed. Please run aria_chk -r on all Aria tables and delete all aria_log.######## files2024-10-27 17:20:35 0 [ERROR] Plugin 'Aria' registration as a STORAGE ENGINE failed.2024-10-27 17:20:35 0 [Note] InnoDB: Mutexes and rw_locks use Windows interlocked functions2024-10-27 17:20:35 0 [Note] InnoDB: Uses event mutexes2024-10-27 17:20:35 0

[Note] InnoDB: Compressed tables use zlib 1.32024-10-27 17:20:35 0 [Note] InnoDB: Number of pools: 12024-10-27 17:20:35 0 [Note] InnoDB: Using SSE2 crc32 instructions2024-10-27 17:20:35 0 [Note] InnoDB: Initializing buffer pool, total size = 16M, instances = 1, chunk size = 16M2024-10-27 17:20:35 0 [Note] InnoDB: Completed initialization of buffer pool2024-10-27 17:20:36 0 [Note] InnoDB: 128 out of 128 rollback segments are active.2024-10-27 17:20:36 0 [Note] InnoDB: Creating shared tablespace for temporary tables2024-10-27 17:20:36 0 [Note] InnoDB: Setting file 'C:\xampp\mysql\data\ibtmp1' size to 12 MB. Physically writing the file full; Please wait ...2024-10-27 17:20:36 0 [Note] InnoDB: File 'C:\xampp\mysql\data\ibtmp1' size is now 12 MB.2024-10-27 17:20:36 0 [Note] InnoDB: Waiting for purge to start2024-10-27 17:20:36 0 [Note] InnoDB: 10.4.32 started; log sequence number 575619; transaction id 4052024-10-27 17:20:36 0 [Note] InnoDB: Loading buffer pool(s) from C:\xampp\mysql\data\ib_buffer_pool2024-10-27 17:20:36 0 [Note] Plugin 'FEEDBACK' is disabled.2024-10-27 17:20:36 0 [ERROR] Could not open mysql.plugin table. Some plugins may be not loaded2024-10-27 17:20:36 0 [ERROR] Failed to initialize plugins.2024-10-27 17:20:36 0 [ERROR] Aborting2024-10-27 17:20:40 0 [Note] Starting MariaDB 10.4.32-MariaDB source revision c4143f909528e3fab0677a28631d10389354c491 as process 21216Cannot find checkpoint record at LSN (1,0x5556)2024-10-27 17:20:40 0 [ERROR] mysqld.exe: Aria recovery failed. Please run aria_chk -r on all Aria tables and delete all aria_log.######## files2024-10-27 17:20:40 0 [ERROR] Plugin 'Aria' registration as a STORAGE ENGINE failed.2024-10-27 17:20:40 0 [Note] InnoDB: Mutexes and rw_locks use Windows interlocked functions2024-10-27 17:20:40 0 [Note] InnoDB: Uses event mutexes2024-10-27 17:20:40 0 [Note] InnoDB: Compressed tables use zlib 1.32024-10-27 17:20:40 0 [Note] InnoDB: Number of pools: 12024-10-27 17:20:40 0 [Note] InnoDB: Using SSE2 crc32 instructions2024-10-27 17:20:40 0 [Note] InnoDB: Initializing buffer pool, total size = 16M, instances = 1, chunk size = 16M2024-10-27 17:20:40 0 [Note] InnoDB: Completed initialization of buffer pool2024-10-27 17:20:40 0 [Note] InnoDB: 128 out of 128 rollback segments are active.2024-10-27 17:20:40 0 [Note] InnoDB: Creating shared tablespace for temporary tables2024-10-27 17:20:40 0 [Note] InnoDB: Setting file 'C:\xampp\mysql\data\ibtmp1' size to 12 MB. Physically writing the file full; Please wait ...2024-10-27 17:20:40 0 [Note] InnoDB: File 'C:\xampp\mysql\data\ibtmp1' size is now 12 MB.2024-10-27 17:20:40 0 [Note] InnoDB: Waiting for purge to start2024-10-27 17:20:40 0 [Note] InnoDB: 10.4.32 started; log sequence number 575628; transaction id 4052024-10-27 17:20:40 0 [Note] InnoDB: Loading buffer pool(s) from C:\xampp\mysql\data\ib_buffer_pool2024-10-27 17:20:40 0 [Note] Plugin 'FEEDBACK' is disabled.2024-10-27 17:20:40 0 [ERROR] Could not open mysql.plugin table. Some plugins may be not loaded2024-10-27 17:20:40 0 [ERROR] Failed to initialize plugins.2024-10-27 17:20:40 0 [ERROR] Aborting2024-10-27 17:20:43 0 [Note] Starting MariaDB 10.4.32-MariaDB source revision c4143f909528e3fab0677a28631d10389354c491 as process 28972Cannot find checkpoint record at LSN (1,0x5556)2024-10-27 17:20:43 0 [ERROR] mysqld.exe: Aria recovery failed. Please run aria_chk -r on all Aria tables and delete all aria_log.######## files2024-10-27 17:20:43 0 [ERROR] Plugin 'Aria' registration as a STORAGE ENGINE failed.2024-10-27 17:20:43 0 [Note] InnoDB: Mutexes and rw_locks use Windows interlocked functions2024-10-27 17:20:43 0 [Note] InnoDB: Uses event mutexes2024-10-27 17:20:43 0 [Note] InnoDB: Compressed tables use zlib 1.32024-10-27 17:20:43 0 [Note] InnoDB: Number of pools: 12024-10-27 17:20:43 0 [Note] InnoDB: Using SSE2 crc32 instructions2024-10-27 17:20:43 0 [Note] InnoDB: Initializing buffer pool, total size = 16M, instances = 1, chunk size = 16M2024-10-27 17:20:43 0 [Note] InnoDB: Completed initialization of buffer pool2024-10-27 17:20:43 0 [Note] InnoDB: 128 out of 128 rollback segments are active.2024-10-27 17:20:43 0 [Note] InnoDB: Creating shared tablespace for temporary tables2024-10-27 17:20:43 0 [Note] InnoDB: Setting file 'C:\xampp\mysql\data\ibtmp1' size to 12 MB. Physically writing the file full; Please wait ...2024-10-27 17:20:43 0 [Note] InnoDB: File 'C:\xampp\mysql\data\ibtmp1' size is now 12 MB.2024-10-27 17:20:43 0 [Note] InnoDB: Waiting for purge to start2024-10-27 17:20:43 0 [Note] InnoDB: 10.4.32 started; log sequence number 575637; transaction id 4052024-10-27 17:20:43 0 [Note] Plugin 'FEEDBACK' is disabled.2024-10-27 17:20:43 0 [ERROR] Could not open mysql.plugin table. Some plugins may be not loaded2024-10-27 17:20:43 0 [Note] InnoDB: Loading buffer pool(s) from C:\xampp\mysql\data\ib_buffer_pool2024-10-27 17:20:43 0 [ERROR] Failed to initialize plugins.2024-10-27 17:20:43 0 [ERROR] Aborting2024-10-27 17:20:49 0 [Note] Starting MariaDB 10.4.32-MariaDB source revision c4143f909528e3fab0677a28631d10389354c491 as process 10784Cannot find checkpoint record at LSN (1,0x5556)2024-10-27 17:20:49 0 [ERROR] mysqld.exe: Aria recovery failed. Please run aria_chk -r on all Aria tables and delete all aria_log.######## files2024-10-27 17:20:49 0 [ERROR] Plugin 'Aria' registration as a STORAGE ENGINE failed.2024-10-27 17:20:49 0 [Note] InnoDB: Mutexes and rw_locks use Windows interlocked functions2024-10-27 17:20:49 0 [Note] InnoDB: Uses event mutexes2024-10-27 17:20:49 0 [Note] InnoDB: Compressed tables use zlib 1.32024-10-27 17:20:49 0 [Note] InnoDB: Number of pools: 12024-10-27 17:20:49 0 [Note] InnoDB: Using SSE2 crc32 instructions2024-10-27 17:20:49 0 [Note] InnoDB: Initializing buffer pool, total size = 16M, instances = 1, chunk size = 16M2024-

10-27 17:20:49 0 [Note] InnoDB: Completed initialization of buffer pool2024-10-27 17:20:49 0 [Note] InnoDB: 128 out of 128 rollback segments are active.2024-10-27 17:20:49 0 [Note] InnoDB: Creating shared tablespace for temporary tables2024-10-27 17:20:49 0 [Note] InnoDB: Setting file 'C:\xampp\mysql\data\ibtmp1' size to 12 MB. Physically writing the file full; Please wait ...2024-10-27 17:20:49 0 [Note] InnoDB: File 'C:\xampp\mysql\data\ibtmp1' size is now 12 MB.2024-10-27 17:20:49 0 [Note] InnoDB: Waiting for purge to start2024-10-27 17:20:49 0 [Note] InnoDB: 10.4.32 started; log sequence number 575646; transaction id 4052024-10-27 17:20:49 0 [Note] InnoDB: Loading buffer pool(s) from C:\xampp\mysql\data\ib_buffer_pool2024-10-27 17:20:49 0 [Note] Plugin 'FEEDBACK' is disabled.2024-10-27 17:20:49 0 [ERROR] Could not open mysql.plugin table. Some plugins may be not loaded2024-10-27 17:20:49 0 [ERROR] Failed to initialize plugins.2024-10-27 17:20:49 0 [ERROR] Aborting

Copilot

Thanks for sharing the logs! It looks like the issue is related to the **Aria storage engine**. Here are the steps to fix it:

1. **Stop the MySQL service** in the XAMPP Control Panel.
2. **Navigate to the MySQL data directory**:
   o Open File Explorer and go to C:\xampp\mysql\data.
3. **Delete the aria_log files**:
   o Look for files named aria_log.######## and delete them.
4. **Run the aria_chk tool** to repair the Aria tables:
   o Open Command Prompt and navigate to the MySQL bin directory: cd C:\xampp\mysql\bin.
   o Run the following command: .\aria_chk -r C:\xampp\mysql\data\mysql*.MAI.

After completing these steps, try restarting the MySQL service in the XAMPP Control Panel. This should resolve the Aria recovery issue.

Does this help? Let me know if you run into any issues or need further assistance!

References

zetbit.tech

You

I tried that and it's still not working, this is the my.ini file:# Example MySQL config file for small systems.## This is for a system with little memory (<= 64M) where MySQL is only used# from time to time and it's important that the mysqld daemon# doesn't use much resources.## You can copy this file to# C:/xampp/mysql/bin/my.cnf to set global options,# mysql-data-dir/my.cnf to set server-specific options (in this# installation this directory is C:/xampp/mysql/data) or# ~/.my.cnf to set user-specific options.## In this file, you can use all long options that a program supports.# If you want to know which options a program supports, run the program# with the "--help" option.# The following options will be passed to all MySQL clients[client]# password = your_password port=3306socket="C:/xampp/mysql/mysql.sock"# Here follows entries for some specific programs # The MySQL serverdefault-character-set=utf8mb4[mysqld]port=3306socket="C:/xampp/mysql/mysql.sock"basedir="C:/xampp/mysql"tmpdir="C:/xampp/tmp"datadir="C:/xampp/mysql/data"pid_file="mysql.pid"# enable-named-pipekey_buffer=16Mmax_allowed_packet=1Msort_buffer_size=512Knet_buffer_length=8Kread_buffer_size=256Kread_rnd_buffer_size=512Kmyisam_sort_buffer_size=8Mlog_error="mysql_error.log"# Change here for bind listening# bind-address="127.0.0.1" # bind-address = ::1 # for ipv6# Where do all the plugins liveplugin_dir="C:/xampp/mysql/lib/plugin/"# Don't listen on a TCP/IP port at all. This can be a security enhancement,# if all processes that need to connect to mysqld run on the same host.# All interaction with mysqld must be made via Unix sockets or named pipes.# Note that using this option without enabling named pipes on Windows# (via the "enable-named-pipe" option) will render mysqld useless!# # commented in by lampp security#skip-networking#skip-federated# Replication Master Server (default)# binary logging is required for replication# log-bin deactivated by default since XAMPP 1.4.11#log-bin=mysql-bin# required unique id between 1 and 2^32 - 1# defaults to 1 if master-host is not set# but will not function as a master if omittedserver-id=1# Replication Slave (comment out master section to use this)## To configure this host as a replication slave, you can choose between# two methods :## 1) Use the CHANGE MASTER TO command (fully described in our manual) -# the syntax is:## CHANGE MASTER TO MASTER_HOST=,

MASTER_PORT=,# MASTER_USER=, MASTER_PASSWORD= ;## where you replace , , by quoted strings and# by the master's port number (3306 by default).## Example:## CHANGE MASTER TO MASTER_HOST='125.564.12.1', MASTER_PORT=3306,# MASTER_USER='joe', MASTER_PASSWORD='secret';## OR## 2) Set the variables below. However, in case you choose this method, then# start replication for the first time (even unsuccessfully, for example# if you mistyped the password in master-password and the slave fails to# connect), the slave will create a master.info file, and any later# change in this file to the variables' values below will be ignored and# overridden by the content of the master.info file, unless you shutdown# the slave server, delete master.info and restart the slaver server.# For that reason, you may want to leave the lines below untouched# (commented) and instead use CHANGE MASTER TO (see above)## required unique id between 2 and 2^32 - 1# (and different from the master)# defaults to 2 if master-host is set# but will not function as a slave if omitted#server-id = 2## The replication master for this slave - required#master-host = ## The username the slave will use for authentication when connecting# to the master - required#master-user = ## The password the slave will authenticate with when connecting to# the master - required#master-password = ## The port the master is listening on.# optional - defaults to 3306#master-port = ## binary logging - not required for slaves, but recommended#log-bin=mysql-bin# Point the following paths to different dedicated disks#tmpdir = "C:/xampp/tmp"#log-update = /path-to-dedicated-directory/hostname# Uncomment the following if you are using BDB tables#bdb_cache_size = 4M#bdb_max_lock = 10000# Comment the following if you are using InnoDB tables#skip-innodbinnodb_data_home_dir="C:/xampp/mysql/data"innodb_data_file_path=ibdata1:10M:autoextendinnodb_log_group_home_dir="C:/xampp/mysql/data"#innodb_log_arch_dir = "C:/xampp/mysql/data"## You can set .._buffer_pool_size up to 50 - 80 %## of RAM but beware of setting memory usage too highinnodb_buffer_pool_size=16M## Set .._log_file_size to 25 % of buffer pool sizeinnodb_log_file_size=5Minnodb_log_buffer_size=8Minnodb_flush_log_at_trx_commit=1innodb_lock_wait_timeout=50## UTF 8 Settings#init-connect='SET NAMES utf8'#collation_server=utf8_unicode_ci#character_set_server=utf8#skip-character-set-client-handshake#character_sets-dir="C:/xampp/mysql/share/charsets"sql_mode=NO_ZERO_IN_DATE,NO_ZERO_DATE,NO_ENGINE_SUBSTITUTIONlog_bin_trust_function_creators=1character-set-server=utf8mb4collation-server=utf8mb4_general_ci[mysqldump]max_allowed_packet=16M[mysql]# Remove the next comment character if you are not familiar with SQL#safe-updates[isamchk]key_buffer=20Msort_buffer_size=20Mread_buffer=2Mwrite_buffer=2M[myisamchk]key_buffer=20Msort_buffer_size=20Mread_buffer=2Mwrite_buffer=2M[mysqlhotcopy]

Copilot

Alright, let's dive deeper. From the logs and the my.ini file, it looks like there's an issue with the Aria storage engine and potentially the InnoDB engine as well.

Here's another approach to fixing the issue:

1. **Backup Your Data**: First, make sure to back up your data directory. Copy the C:\xampp\mysql\data directory to a safe location.
2. **Delete Specific Logs and Files**:
   o Go to C:\xampp\mysql\data.
   o Delete aria_log_control and aria_log.######## files.
   o Delete ibtmp1 and ib_logfile0 and ib_logfile1.
3. **Modify my.ini**:
   o Open the my.ini file again and add the following lines under [mysqld]:
innodb_force_recovery = 1
      This option forces InnoDB to start up in recovery mode.
4. **Restart MySQL**:
   o Try to start MySQL from the XAMPP Control Panel.
5. **Repair Tables**:
   o If MySQL starts successfully, run the following command to repair your Aria tables:
      ▪ Open Command Prompt and navigate to the MySQL bin directory: cd C:\xampp\mysql\bin.
      ▪ Run this command: .\aria_chk -r C:\xampp\mysql\data\mysql*.MAI.
6. **Remove Recovery Mode**:
   o Once everything is up and running, remember to remove the innodb_force_recovery line from your my.ini file and restart MySQL again.

These steps should help you get your MySQL service running again. If you run into any further issues or need more detailed guidance, feel free to ask!

You

I've never used mySQL before, would it work if I just delet all the associated files then re-run mySQL?

Copilot

Hmm. Deleting all the associated files might get you back up and running, but you'd lose any data you haven't backed up. It's more like hitting the reset button. If you don't have anything vital in there, it's an option. Another possible route is uninstalling and then reinstalling MySQL within XAMPP. This way, you get a fresh start without manually deleting files. Still, back up anything important first. Want to give that a go?

You

How do I go about uninstalling it from xampp and then reinstalling?

Copilot

No worries, I'll guide you through the process step-by-step:

Uninstalling XAMPP:

1. **Stop Services**: Open the XAMPP Control Panel and stop both Apache and MySQL services.

2. **Uninstall XAMPP**: Go to Control Panel > Programs > Programs and Features. Find XAMPP in the list, select it, and click Uninstall. Follow the prompts to complete the uninstallation.

3. **Delete Remaining Files**: Navigate to C:\xampp and delete the entire XAMPP folder.

Reinstalling XAMPP:

1. **Download XAMPP**: Go to the official XAMPP website and download the latest version.

2. **Install XAMPP**: Run the installer and follow the prompts. Make sure to select all the required components during the installation process.

3. **Restore Data**: If you have any important data from your previous installation, you can restore it now. For MySQL data, copy the C:\xampp\mysql\data folder from your backup to the new XAMPP installation directory.

After reinstalling, open the XAMPP Control Panel and start the MySQL service. This should give you a fresh start with MySQL.

Does this help? Let me know if you run into any issues or need further assistance!

Learn more: 1. oreilly.com 2. www.youtube.com 3. tutsmake.com 4. stellarinfo.com 5. communit.apachefriends.org

You

(Now Monday, Oct 28)
How can I write a somewhat formal email to my work colleagues regarding these 3 main points:

- I'm now back at the Trust, remember I was working on that AI document extraction program before
- Now I've started working on it again, I think it should be ready for demonstration in 2 or 3 months
- I'll give an update of my progress after about a month

Copilot

Sure thing! Here's a draft you might find useful:

Subject: Resumption of AI Document Extraction Program

Dear Team,

I hope this message finds you well.

I am pleased to inform you that I have resumed my work at the Trust. You may recall that before my departure, I was developing an AI document extraction program. I have now returned to this project, and I anticipate that it should be ready for demonstration within the next two to three months.

I will provide an update on my progress in approximately one month.

Thank you for your continued support.

Best regards,
[Your Name]

This should cover all the key points while keeping it professional and to the point. How does it look?

You

How can I adapt this as an email only addressing Katharine Foxton

Copilot

Got it. Here's a tailored email for Katharine:

Subject: Resumption of AI Document Extraction Program

Hi Katharine,
I hope this message finds you well.
I wanted to let you know that I'm back at the Trust. You might remember that before my departure, I was working on the AI document extraction program. I've now resumed work on it and expect it should be ready for demonstration within the next two to three months.
I will provide an update on my progress in about a month.
Thank you for your continued support.
Best regards,
[Your Name]

This version keeps it direct and personal. Thoughts?

You

I meant as a Teams message

Copilot

Got it. Here's a Teams message you could send to Katharine:

Hi Katharine,
Just letting you know I'm back at the Trust and have resumed work on the AI document extraction program. I expect it will be ready for demonstration in the next two to three months. I'll provide an update on my progress in about a month.
Thanks for your continued support!
Best regards,
[Your Name]

Does that work?

You

I have the following python code, how can I redirect back to index when a file is successfully uploaded and then output the message:

```
from flask import Flask, request, redirect, url_for, render_template
import os

app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = 'localFiles'
os.makedirs(app.config['UPLOAD_FOLDER'], exist_ok=True)


@app.route('/')
def index():
    return render_template('index.html')


@app.route('/upload', methods=['POST'])
def upload_file():
    if 'file' not in request.files:
        return 'No file part'
    file = request.files['file']
    if file.filename == '':
        return 'No selected file'
    if file:
        filename = file.filename
        file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
        return 'File successfully uploaded'


if __name__ == '__main__':
    app.run(debug=True)
```

Copilot

To redirect back to the index and display a success message, you can use Flask's flash method for messages and redirect function for redirection. Here's an updated version of your code:

```
from flask import Flask, request, redirect, url_for, render_template, flash
import os

app = Flask(__name__)
app.secret_key = 'supersecretkey'  # Required for flash messages
app.config['UPLOAD_FOLDER'] = 'localFiles'
os.makedirs(app.config['UPLOAD_FOLDER'], exist_ok=True)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/upload', methods=['POST'])
def upload_file():
    if 'file' not in request.files:
        flash('No file part')
        return redirect(url_for('index'))

    file = request.files['file']
    if file.filename == '':
        flash('No selected file')
        return redirect(url_for('index'))

    if file:
        filename = file.filename
        file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
        flash('File successfully uploaded')
        return redirect(url_for('index'))
```

```
if __name__ == '__main__':
    app.run(debug=True)
```

In your index.html template, you'll need to display the flashed messages:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Index</title>
</head>
<body>
    {% with messages = get_flashed_messages() %}
        {% if messages %}
            <ul>
                {% for message in messages %}
                    <li>{{ message }}</li>
                {% endfor %}
            </ul>
        {% endif %}
    {% endwith %}
    <!-- Form and other content goes here -->
</body>
</html>
```

This setup will redirect back to the index page and display the success message when a file is successfully uploaded. Let me know if you need any further tweaks!

References

1. github.com 2. github.com 3. github.com 4. github.com 5. github.com 6. github.com

You

This is my index template webpage, how can I update it:
```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>File Upload</title>
    <style>
      .container {
        width: 400px;
        margin: 0 auto;
        text-align: center;
        padding: 20px;
        border: 1px solid #ddd;
        border-radius: 8px;
        box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
      }
      h1 {
        font-size: 2em;
        margin-bottom: 20px;
      }
      label {
        display: block;
        margin: 15px 0 5px;
        font-size: 1.2em;
      }
      input[type="file"], input[type="text"] {
        width: 100%;
        padding: 10px;
        margin-bottom: 20px;
        border: 1px solid #ccc;
        border-radius: 4px;
      }
```

```css
    button {
      padding: 10px 20px;
      background-color: #4CAF50;
      color: white;
      border: none;
      border-radius: 4px;
      cursor: pointer;
    }
    button:hover {
      background-color: #45a049;
    }
  </style>
</head>
<body>
  <div class="container">
    <h1>File Upload</h1>
    <form action="/upload" method="post" enctype="multipart/form-data">
      <label for="file-upload">File Upload</label>
      <input type="file" id="file-upload" name="file" accept=".doc, .docx">
      <label for="output-file-name">Output File Name</label>
      <input type="text" id="output-file-name" name="output-filename" placeholder="Enter output file name">
      <button type="submit">Upload</button>
    </form>
  </div>
</body>
</html>
```

Copilot

Sure thing! Here's an updated version of your index template to include the flash messages for feedback after file upload:

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>File Upload</title>
  <style>
    .container {
      width: 400px;
      margin: 0 auto;
      text-align: center;
      padding: 20px;
      border: 1px solid #ddd;
      border-radius: 8px;
      box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    }
    h1 {
      font-size: 2em;
      margin-bottom: 20px;
    }
    label {
      display: block;
      margin: 15px 0 5px;
      font-size: 1.2em;
    }
    input[type="file"],
    input[type="text"] {
      width: 100%;
      padding: 10px;
      margin-bottom: 20px;
      border: 1px solid #ccc;
      border-radius: 4px;
    }
    button {
      padding: 10px 20px;
      background-color: #4CAF50;
      color: white;
      border: none;
      border-radius: 4px;
      cursor: pointer;
    }
```

```
        button:hover {
            background-color: #45a049;
        }
        .flash-message {
            margin-bottom: 20px;
            padding: 10px;
            border: 1px solid #4CAF50;
            background-color: #d4edda;
            color: #155724;
            border-radius: 4px;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>File Upload</h1>
        {% with messages = get_flashed_messages() %}
            {% if messages %}
                <div class="flash-message">
                    {% for message in messages %}
                        <p>{{ message }}</p>
                    {% endfor %}
                </div>
            {% endif %}
        {% endwith %}
        <form action="/upload" method="post" enctype="multipart/form-data">
            <label for="file-upload">File Upload</label>
            <input type="file" id="file-upload" name="file" accept=".doc, .docx">
            <label for="output-file-name">Output File Name</label>
            <input type="text" id="output-file-name" name="output-filename" placeholder="Enter output file name">
            <button type="submit">Upload</button>
        </form>
    </div>
</body>
</html>
```

This includes a div to display flash messages, which will show feedback after a file is uploaded. Let me know if you need anything else!

You

I want to put the flash messages after evrything else

Copilot

Got it! Here's an updated version of your index template with the flash messages placed after everything else:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>File Upload</title>
    <style>
        .container {
            width: 400px;
            margin: 0 auto;
            text-align: center;
            padding: 20px;
            border: 1px solid #ddd;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }
        h1 {
            font-size: 2em;
            margin-bottom: 20px;
        }
        label {
            display: block;
            margin: 15px 0 5px;
            font-size: 1.2em;
        }
```

```
        input[type="file"],
        input[type="text"] {
            width: 100%;
            padding: 10px;
            margin-bottom: 20px;
            border: 1px solid #ccc;
            border-radius: 4px;
        }
        button {
            padding: 10px 20px;
            background-color: #4CAF50;
            color: white;
            border: none;
            border-radius: 4px;
            cursor: pointer;
        }
        button:hover {
            background-color: #45a049;
        }
        .flash-message {
            margin-top: 20px;
            padding: 10px;
            border: 1px solid #4CAF50;
            background-color: #d4edda;
            color: #155724;
            border-radius: 4px;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>File Upload</h1>
        <form action="/upload" method="post" enctype="multipart/form-data">
            <label for="file-upload">File Upload</label>
            <input type="file" id="file-upload" name="file" accept=".doc, .docx">
            <label for="output-file-name">Output File Name</label>
            <input type="text" id="output-file-name" name="output-filename" placeholder="Enter output file name">
            <button type="submit">Upload</button>
        </form>
        {% with messages = get_flashed_messages() %}
            {% if messages %}
                <div class="flash-message">
                    {% for message in messages %}
                        <p>{{ message }}</p>
                    {% endfor %}
                </div>
            {% endif %}
        {% endwith %}
    </div>
</body>
</html>
```

This way, the flash messages appear after the form content. Let me know if this works better!

You

I want to display "File not uploaded yet" as the message at teh beginning

Copilot

Got it. Here's your updated template that displays "File not uploaded yet" at the beginning, and then shows the flash messages after everything else:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>File Upload</title>
    <style>
        .container {
            width: 400px;
```

```
            margin: 0 auto;
            text-align: center;
            padding: 20px;
            border: 1px solid #ddd;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }
        h1 {
            font-size: 2em;
            margin-bottom: 20px;
        }
        label {
            display: block;
            margin: 15px 0 5px;
            font-size: 1.2em;
        }
        input[type="file"],
        input[type="text"] {
            width: 100%;
            padding: 10px;
            margin-bottom: 20px;
            border: 1px solid #ccc;
            border-radius: 4px;
        }
        button {
            padding: 10px 20px;
            background-color: #4CAF50;
            color: white;
            border: none;
            border-radius: 4px;
            cursor: pointer;
        }
        button:hover {
            background-color: #45a049;
        }
        .flash-message {
            margin-top: 20px;
            padding: 10px;
            border: 1px solid #4CAF50;
            background-color: #d4edda;
            color: #155724;
            border-radius: 4px;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>File Upload</h1>
        <div class="flash-message">
            <p>File not uploaded yet</p>
        </div>
        <form action="/upload" method="post" enctype="multipart/form-data">
            <label for="file-upload">File Upload</label>
            <input type="file" id="file-upload" name="file" accept=".doc, .docx">
            <label for="output-file-name">Output File Name</label>
            <input type="text" id="output-file-name" name="output-filename" placeholder="Enter output file name">
            <button type="submit">Upload</button>
        </form>
        {% with messages = get_flashed_messages() %}
            {% if messages %}
                <div class="flash-message">
                    {% for message in messages %}
                        <p>{{ message }}</p>
                    {% endfor %}
                </div>
            {% endif %}
        {% endwith %}
    </div>
</body>
</html>
```

With this setup, the page initially displays "File not uploaded yet" and then shows any subsequent flash messages below the form. How does this look to you?

You

No I mean before the user uploads a file, to show this instead of the flash messages

Copilot

Got it. To display "File not uploaded yet" initially and then replace it with flash messages after an upload, you can use a bit of template logic. Here's an updated version of your index template:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>File Upload</title>
  <style>
    .container {
      width: 400px;
      margin: 0 auto;
      text-align: center;
      padding: 20px;
      border: 1px solid #ddd;
      border-radius: 8px;
      box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    }
    h1 {
      font-size: 2em;
      margin-bottom: 20px;
    }
    label {
      display: block;
      margin: 15px 0 5px;
      font-size: 1.2em;
    }
    input[type="file"],
    input[type="text"] {
      width: 100%;
      padding: 10px;
      margin-bottom: 20px;
      border: 1px solid #ccc;
      border-radius: 4px;
    }
    button {
      padding: 10px 20px;
      background-color: #4CAF50;
      color: white;
      border: none;
      border-radius: 4px;
      cursor: pointer;
    }
    button:hover {
      background-color: #45a049;
    }
    .flash-message {
      margin-top: 20px;
      padding: 10px;
      border: 1px solid #4CAF50;
      background-color: #d4edda;
      color: #155724;
      border-radius: 4px;
    }
  }
  </style>
```

```
</head>
<body>
  <div class="container">
    <h1>File Upload</h1>
    <form action="/upload" method="post" enctype="multipart/form-data">
      <label for="file-upload">File Upload</label>
      <input type="file" id="file-upload" name="file" accept=".doc, .docx">
      <label for="output-file-name">Output File Name</label>
      <input type="text" id="output-file-name" name="output-filename" placeholder="Enter output file name">
      <button type="submit">Upload</button>
    </form>
    {% with messages = get_flashed_messages() %}
      {% if messages %}
        <div class="flash-message">
          {% for message in messages %}
            <p>{{ message }}</p>
          {% endfor %}
        </div>
      {% else %}
        <div class="flash-message">
          <p>File not uploaded yet</p>
        </div>
      {% endif %}
    {% endwith %}
  </div>
</body>
</html>
```

This way, "File not uploaded yet" will be shown initially, and any flash messages from the upload process will replace it. Let me know if this helps!

You

In my Python code, I want it to actually save the uploaded file to the localFiles folder and keep it there so that it can be used afterwards, I also want a check for if the file already exists in localFiles and then put a flash message that reads "File already uploaded" and not do anything else then

Copilot

Here's an updated version of your Python code to check if the file already exists in the localFiles folder and flash the appropriate message:

```
from flask import Flask, request, redirect, url_for, render_template, flash
import os

app = Flask(__name__)
app.secret_key = 'supersecretkey'  # Required for flash messages
app.config['UPLOAD_FOLDER'] = 'localFiles'
os.makedirs(app.config['UPLOAD_FOLDER'], exist_ok=True)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/upload', methods=['POST'])
def upload_file():
    if 'file' not in request.files:
        flash('No file part')
        return redirect(url_for('index'))

    file = request.files['file']
    if file.filename == '':
        flash('No selected file')
        return redirect(url_for('index'))
```

```
    if file:
        filename = file.filename
        file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        if os.path.exists(file_path):
            flash('File already uploaded')
        else:
            file.save(file_path)
            flash('File successfully uploaded')

    return redirect(url_for('index'))

if __name__ == '__main__':
    app.run(debug=True)
```

This version includes a check to see if the file already exists in the localFiles directory before saving it. If the file exists, it flashes the message "File already uploaded" and does nothing else. Otherwise, it saves the file and flashes "File successfully uploaded".

Let me know how it works for you!

References

1. github.com 2. github.com 3. github.com

You

How can I include an "Extract Info" button on the html on the same line as teh "upload

Copilot

You can include the "Extract Info" button on the same line as the "Upload" button by using a bit of CSS to style the buttons and ensure they sit next to each other. Here's how you can update your HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>File Upload</title>
    <style>
        .container {
            width: 400px;
            margin: 0 auto;
            text-align: center;
            padding: 20px;
            border: 1px solid #ddd;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }
        h1 {
            font-size: 2em;
            margin-bottom: 20px;
        }
        label {
            display: block;
            margin: 15px 0 5px;
            font-size: 1.2em;
        }
        input[type="file"],
        input[type="text"] {
            width: 100%;
            padding: 10px;
            margin-bottom: 20px;
            border: 1px solid #ccc;
```

```
        border-radius: 4px;
      }
      .button-container {
        display: flex;
        justify-content: center;
        gap: 10px;
      }
      button {
        padding: 10px 20px;
        background-color: #4CAF50;
        color: white;
        border: none;
        border-radius: 4px;
        cursor: pointer;
      }
      button:hover {
        background-color: #45a049;
      }
      .flash-message {
        margin-top: 20px;
        padding: 10px;
        border: 1px solid #4CAF50;
        background-color: #d4edda;
        color: #155724;
        border-radius: 4px;
      }
    </style>
  </head>
  <body>
    <div class="container">
      <h1>File Upload</h1>
      <form action="/upload" method="post" enctype="multipart/form-data">
        <label for="file-upload">File Upload</label>
        <input type="file" id="file-upload" name="file" accept=".doc, .docx">
        <label for="output-file-name">Output File Name</label>
        <input type="text" id="output-file-name" name="output-filename" placeholder="Enter output file name">
        <div class="button-container">
          <button type="submit">Upload</button>
          <button type="button" onclick="extractInfo()">Extract Info</button>
        </div>
      </form>
      {% with messages = get_flashed_messages() %}
        {% if messages %}
          <div class="flash-message">
            {% for message in messages %}
              <p>{{ message }}</p>
            {% endfor %}
          </div>
        {% else %}
          <div class="flash-message">
            <p>File not uploaded yet</p>
          </div>
        {% endif %}
      {% endwith %}
    </div>
    <script>
      function extractInfo() {
        // Logic for extracting information goes here
        alert('Extracting information...');
      }
    </script>
  </body>
</html>
```

This code places the "Extract Info" button next to the "Upload" button using a flexbox container, ensuring they are aligned on the same line. You can add the logic for the "Extract Info" functionality in the extractInfo function. Is this what you were looking for?

You

Yes, but I didn't finish my sentence, I want the "Extract Info" button to link back to the Python Flask code and to define a method in there called "process_and_download" that calls another method [already defined] called documentExtractor.extractInfo(filename) and then saves the output file using the file name obtained from the input with id "output-file-name" and also using outFolder as the folder so the new file path should be "out_file_path = os.path.join(outFolder, outputFileName)". I also want to download the saved file through the web browser. What would be the full updated html template?

Copilot

Okay, here's the updated HTML template that includes the "Extract Info" button and links back to the Flask app. This will call the process_and_download method in your Flask app, which uses the documentExtractor.extractInfo(filename) method and saves the output file.

Updated HTML Template:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>File Upload</title>
  <style>
    .container {
      width: 400px;
      margin: 0 auto;
      text-align: center;
      padding: 20px;
      border: 1px solid #ddd;
      border-radius: 8px;
      box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    }
    h1 {
      font-size: 2em;
      margin-bottom: 20px;
    }
    label {
      display: block;
      margin: 15px 0 5px;
      font-size: 1.2em;
    }
    input[type="file"],
    input[type="text"] {
      width: 100%;
      padding: 10px;
      margin-bottom: 20px;
      border: 1px solid #ccc;
      border-radius: 4px;
    }
    .button-container {
      display: flex;
      justify-content: center;
      gap: 10px;
    }
    button {
      padding: 10px 20px;
      background-color: #4CAF50;
      color: white;
      border: none;
      border-radius: 4px;
```

```css
            cursor: pointer;
        }
        button:hover {
            background-color: #45a049;
        }
        .flash-message {
            margin-top: 20px;
            padding: 10px;
            border: 1px solid #4CAF50;
            background-color: #d4edda;
            color: #155724;
            border-radius: 4px;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>File Upload</h1>
        <form action="/upload" method="post" enctype="multipart/form-data">
            <label for="file-upload">File Upload</label>
            <input type="file" id="file-upload" name="file" accept=".doc, .docx">
            <label for="output-file-name">Output File Name</label>
            <input type="text" id="output-file-name" name="output-filename" placeholder="Enter output file name">
            <div class="button-container">
                <button type="submit">Upload</button>
                <button type="button" onclick="extractInfo()">Extract Info</button>
            </div>
        </form>
        {% with messages = get_flashed_messages() %}
            {% if messages %}
                <div class="flash-message">
                    {% for message in messages %}
                        <p>{{ message }}</p>
                    {% endfor %}
                </div>
            {% else %}
                <div class="flash-message">
                    <p>File not uploaded yet</p>
                </div>
            {% endif %}
        {% endwith %}
    </div>
    <script>
        function extractInfo() {
            const filename = document.getElementById('file-upload').files[0]?.name;
            const outputFileName = document.getElementById('output-file-name').value;
            if (filename && outputFileName) {
                window.location.href =
`/process_and_download?filename=${filename}&outputFileName=${outputFileName}`;
            } else {
                alert('Please upload a file and specify an output file name.');
            }
        }
    </script>
</body>
</html>
```

Updated Flask Code:

```python
from flask import Flask, request, redirect, url_for, render_template, flash, send_file
import os
import documentExtractor

app = Flask(__name__)
```

```python
app.secret_key = 'supersecretkey'  # Required for flash messages
app.config['UPLOAD_FOLDER'] = 'localFiles'
app.config['OUTPUT_FOLDER'] = 'outFolder'
os.makedirs(app.config['UPLOAD_FOLDER'], exist_ok=True)
os.makedirs(app.config['OUTPUT_FOLDER'], exist_ok=True)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/upload', methods=['POST'])
def upload_file():
    if 'file' not in request.files:
        flash('No file part')
        return redirect(url_for('index'))

    file = request.files['file']
    if file.filename == '':
        flash('No selected file')
        return redirect(url_for('index'))

    if file:
        filename = file.filename
        file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        if os.path.exists(file_path):
            flash('File already uploaded')
        else:
            file.save(file_path)
            flash('File successfully uploaded')

    return redirect(url_for('index'))

@app.route('/process_and_download')
def process_and_download():
    filename = request.args.get('filename')
    output_filename = request.args.get('outputFileName')
    if not filename or not output_filename:
        flash('Missing filename or output file name')
        return redirect(url_for('index'))

    input_file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
    output_file_path = os.path.join(app.config['OUTPUT_FOLDER'], output_filename)

    # Call the documentExtractor to process the file
    documentExtractor.extractInfo(input_file_path)

    # Save the output file
    # Assuming extractInfo method modifies the file and saves it as output_file_path
    with open(output_file_path, 'w') as f:
        f.write('Processed content')  # Replace with actual content from extractInfo method

    return send_file(output_file_path, as_attachment=True)

if __name__ == '__main__':
    app.run(debug=True)
```

This setup ensures that the "Extract Info" button calls the process_and_download method, which processes the file using documentExtractor.extractInfo(filename), saves the output file, and then initiates the download through the web browser. How does this look to you?

References

1. github.com 2. github.com

You

How can I use regular expressions to get a string of everything in a file path except the filename e.g. the regular expression should get "https://github.com/HasanBradfordUni/BCFT-AI/blob/main/Other%20AI%20Projects/AI%20Document%20Extraction/doc/blank/" if the file path is "https://github.com/HasanBradfordUni/BCFT-AI/blob/main/Other%20AI%20Projects/AI%20Document%20Extraction/doc/blank/Case%20management%20profile%20-%20blank.docx" and then I want to use string.replace() method to replace it with "localFiles/" so the final path that I want is then "localFiles/Case%20management%20profile%20-%20blank.docx"?

Copilot

You can use regular expressions to achieve this. Here's how you can extract the path excluding the filename and then replace it with localFiles/:

```
import re

# Your file path
file_path = "https://github.com/HasanBradfordUni/BCFT-AI/blob/main/Other%20AI%20Projects/AI%20Document%20Extraction/doc/blank/Case%20management%20profile%20-%20blank.docx"

# Regular expression to match everything before the filename
pattern = r"(.*/)"

# Extract the directory path
match = re.match(pattern, file_path)
if match:
    directory_path = match.group(1)
    # Replace the directory path with "localFiles/"
    new_path = file_path.replace(directory_path, "localFiles/")
    print(new_path)  # Output: localFiles/Case%20management%20profile%20-%20blank.docx
else:
    print("No match found")
```

In this code, the regular expression r"(.*/)" matches everything up to and including the last / in the file path, capturing it as a group. We then use the re.match function to apply this pattern to the file path. If a match is found, the captured group (the directory path) is replaced with "localFiles/" using the replace method.

Give it a try and see if it works for you!

You

My process and download isn't working as intended, I want to download the file at the end and then redirect back to index, here is teh relevant code:

```
@app.route('/process', methods=['POST'])
def process_and_download():
 filename = request.form['file']
 output_filename = request.form['output-filename']
 file_path = os.path.join(folderName, filename)
 if not os.path.exists(file_path):
   flash('File not found')
   return redirect(url_for('index'))
 # Assuming documentExtractor is defined and available
 mapped_data = documentExtractor.extractInfo(file_path)
 out_file_path = documentExtractor.save(output_filename, mapped_data)
 return send_file(out_file_path, as_attachment=True)
```

Relevant code from documentExtractor:

```
def localiseFilePath(self, file_path):
 # Regular expression to match everything before the filename
 pattern = r"(.*/)"
```

```python
    new_folder_path = 'Other AI Projects/AI Document Extraction/src/localFiles/'
    # Extract the folder path
    match = re.match(pattern, file_path)
    if match:
      directory_path = match.group(1)
      # Replace the previous folder path with the new folder path
      new_path = file_path.replace(directory_path, new_folder_path)
      return new_path
    else:
      print("No match found")
      return "./"


def save(self, out_file_path, mapped_data):
    out_file = self.localiseFilePath(out_file_path)
    try:
      output_file = self.insert_data_into_docx(self.input_file, out_file, mapped_data)
      print(f"File {output_file} saved successfully!")
    except:
      print(f"File {output_file} could not be saved! Unknown error!")
      return out_file
```

Copilot

I see the challenge. To both download the file and then redirect back to the index, you need a two-step process. The file download will be handled by a specific route, while the redirection will be handled separately. Here's how you can adjust your code to achieve this:

Updated Flask Code:

```python
from flask import Flask, request, redirect, url_for, render_template, flash, send_file, make_response
import os
import documentExtractor


app = Flask(__name__)
app.secret_key = 'supersecretkey'  # Required for flash messages
app.config['UPLOAD_FOLDER'] = 'localFiles'
app.config['OUTPUT_FOLDER'] = 'outFolder'
os.makedirs(app.config['UPLOAD_FOLDER'], exist_ok=True)
os.makedirs(app.config['OUTPUT_FOLDER'], exist_ok=True)


@app.route('/')
def index():
    return render_template('index.html')


@app.route('/upload', methods=['POST'])
def upload_file():
    if 'file' not in request.files:
        flash('No file part')
        return redirect(url_for('index'))

    file = request.files['file']
    if file.filename == '':
        flash('No selected file')
        return redirect(url_for('index'))

    if file:
        filename = file.filename
        file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        if os.path.exists(file_path):
            flash('File already uploaded')
        else:
            file.save(file_path)
            flash('File successfully uploaded')
```

```python
    return redirect(url_for('index'))

@app.route('/process', methods=['POST'])
def process_and_download():
    filename = request.form['file']
    output_filename = request.form['output-filename']
    file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)

    if not os.path.exists(file_path):
        flash('File not found')
        return redirect(url_for('index'))

    # Assuming documentExtractor is defined and available
    mapped_data = documentExtractor.extractInfo(file_path)
    out_file_path = os.path.join(app.config['OUTPUT_FOLDER'], output_filename)

    # Save the output file
    documentExtractor.save(out_file_path, mapped_data)

    response = make_response(redirect(url_for('download_file', filepath=out_file_path)))
    response.headers['Location'] += "?redirect_after=true"
    return response

@app.route('/download')
def download_file():
    filepath = request.args.get('filepath')
    redirect_after = request.args.get('redirect_after')

    if not os.path.exists(filepath):
        flash('File not found')
        return redirect(url_for('index'))

    response = make_response(send_file(filepath, as_attachment=True))

    if redirect_after:
        response.headers['Refresh'] = '1;url=/'

    return response

if __name__ == '__main__':
    app.run(debug=True)
```

Explanation:

1. **process_and_download route**: This handles the processing and then redirects to the download route, appending a redirect_after=true parameter.

2. **download_file route**: This handles the file download. If the redirect_after parameter is present, it adds a refresh header to redirect back to the index page after the download.

This should give you the desired behavior: processing the file, downloading it, and then redirecting back to the index page. Try it out and see if it works for you!

References

1. github.com 2. github.com

You

Does the index template webpage need to be updated for this to work?

Copilot

The current version of your index template should work with the updated Flask code. However, to handle the extraction process through a form submission, you'll need to update the form to submit to the /process route. Here's your updated index template with the necessary changes:

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>File Upload</title>
  <style>
    .container {
      width: 400px;
      margin: 0 auto;
      text-align: center;
      padding: 20px;
      border: 1px solid #ddd;
      border-radius: 8px;
      box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    }
    h1 {
      font-size: 2em;
      margin-bottom: 20px;
    }
    label {
      display: block;
      margin: 15px 0 5px;
      font-size: 1.2em;
    }
    input[type="file"],
    input[type="text"] {
      width: 100%;
      padding: 10px;
      margin-bottom: 20px;
      border: 1px solid #ccc;
      border-radius: 4px;
    }
    .button-container {
      display: flex;
      justify-content: center;
      gap: 10px;
    }
    button {
      padding: 10px 20px;
      background-color: #4CAF50;
      color: white;
      border: none;
      border-radius: 4px;
      cursor: pointer;
    }
    button:hover {
      background-color: #45a049;
    }
    .flash-message {
      margin-top: 20px;
      padding: 10px;
      border: 1px solid #4CAF50;
      background-color: #d4edda;
      color: #155724;
      border-radius: 4px;
    }
  </style>
</head>
```

```html
<body>
    <div class="container">
        <h1>File Upload</h1>
        <form id="upload-form" action="/upload" method="post" enctype="multipart/form-data">
            <label for="file-upload">File Upload</label>
            <input type="file" id="file-upload" name="file" accept=".doc, .docx">
            <label for="output-file-name">Output File Name</label>
            <input type="text" id="output-file-name" name="output-filename" placeholder="Enter output file name">
            <div class="button-container">
                <button type="submit">Upload</button>
                <button type="button" onclick="submitProcessForm()">Extract Info</button>
            </div>
        </form>
        <form id="process-form" action="/process" method="post" style="display: none;">
            <input type="hidden" id="hidden-file" name="file">
            <input type="hidden" id="hidden-output-file-name" name="output-filename">
        </form>
        {% with messages = get_flashed_messages() %}
            {% if messages %}
                <div class="flash-message">
                    {% for message in messages %}
                        <p>{{ message }}</p>
                    {% endfor %}
                </div>
            {% else %}
                <div class="flash-message">
                    <p>File not uploaded yet</p>
                </div>
            {% endif %}
        {% endwith %}
    </div>
    <script>
        function submitProcessForm() {
            const fileInput = document.getElementById('file-upload');
            const outputFileNameInput = document.getElementById('output-file-name');
            const hiddenFileInput = document.getElementById('hidden-file');
            const hiddenOutputFileNameInput = document.getElementById('hidden-output-file-name');

            if (fileInput.files.length > 0 && outputFileNameInput.value) {
                hiddenFileInput.value = fileInput.files[0].name;
                hiddenOutputFileNameInput.value = outputFileNameInput.value;
                document.getElementById('process-form').submit();
            } else {
                alert('Please upload a file and specify an output file name.');
            }
        }
    </script>
</body>
</html>
```

Explanation:

1. **Two Forms**: There are now two forms on the page. The first form (upload-form) handles file uploads. The second form (process-form) is hidden and will be used to submit the file name and output file name to the /process route.

2. **JavaScript Function**: The submitProcessForm() function is triggered when the "Extract Info" button is clicked. It checks if a file is selected and an output file name is provided, then fills the hidden inputs in the process-form and submits it.

This setup should work seamlessly with the updated Flask code to handle file uploads, processing, and downloading. Give it a try and let me know how it goes!

I've done that but the file doesn't download, everything else works fine

Let's fine-tune it. Here's an updated approach to ensure the file gets downloaded:

Flask Code:

```
from flask import Flask, request, redirect, url_for, render_template, flash, send_file, make_response
import os
import documentExtractor

app = Flask(__name__)
app.secret_key = 'supersecretkey'  # Required for flash messages
app.config['UPLOAD_FOLDER'] = 'localFiles'
app.config['OUTPUT_FOLDER'] = 'outFolder'
os.makedirs(app.config['UPLOAD_FOLDER'], exist_ok=True)
os.makedirs(app.config['OUTPUT_FOLDER'], exist_ok=True)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/upload', methods=['POST'])
def upload_file():
    if 'file' not in request.files:
        flash('No file part')
        return redirect(url_for('index'))

    file = request.files['file']
    if file.filename == '':
        flash('No selected file')
        return redirect(url_for('index'))

    if file:
        filename = file.filename
        file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        if os.path.exists(file_path):
            flash('File already uploaded')
        else:
            file.save(file_path)
            flash('File successfully uploaded')

    return redirect(url_for('index'))

@app.route('/process', methods=['POST'])
def process_and_download():
    filename = request.form['file']
    output_filename = request.form['output-filename']
    file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)

    if not os.path.exists(file_path):
        flash('File not found')
        return redirect(url_for('index'))

    # Assuming documentExtractor is defined and available
    mapped_data = documentExtractor.extractInfo(file_path)
    out_file_path = os.path.join(app.config['OUTPUT_FOLDER'], output_filename)

    # Save the output file
    documentExtractor.save(out_file_path, mapped_data)

    return redirect(url_for('download_file', filepath=out_file_path))

@app.route('/download')
def download_file():
    filepath = request.args.get('filepath')
```

```python
    if not os.path.exists(filepath):
        flash('File not found')
        return redirect(url_for('index'))

    return send_file(filepath, as_attachment=True)

if __name__ == '__main__':
    app.run(debug=True)
```

HTML Template:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>File Upload</title>
    <style>
        .container {
            width: 400px;
            margin: 0 auto;
            text-align: center;
            padding: 20px;
            border: 1px solid #ddd;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }
        h1 {
            font-size: 2em;
            margin-bottom: 20px;
        }
        label {
            display: block;
            margin: 15px 0 5px;
            font-size: 1.2em;
        }
        input[type="file"],
        input[type="text"] {
            width: 100%;
            padding: 10px;
            margin-bottom: 20px;
            border: 1px solid #ccc;
            border-radius: 4px;
        }
        .button-container {
            display: flex;
            justify-content: center;
            gap: 10px;
        }
        button {
            padding: 10px 20px;
            background-color: #4CAF50;
            color: white;
            border: none;
            border-radius: 4px;
            cursor: pointer;
        }
        button:hover {
            background-color: #45a049;
        }
        .flash-message {
            margin-top: 20px;
            padding: 10px;
            border: 1px solid #4CAF50;
            background-color: #d4edda;
            color: #155724;
```

```html
      border-radius: 4px;
    }
  </style>
</head>
<body>
  <div class="container">
    <h1>File Upload</h1>
    <form id="upload-form" action="/upload" method="post" enctype="multipart/form-data">
      <label for="file-upload">File Upload</label>
      <input type="file" id="file-upload" name="file" accept=".doc, .docx">
      <label for="output-file-name">Output File Name</label>
      <input type="text" id="output-file-name" name="output-filename" placeholder="Enter output file name">
      <div class="button-container">
        <button type="submit">Upload</button>
        <button type="button" onclick="submitProcessForm()">Extract Info</button>
      </div>
    </form>
    <form id="process-form" action="/process" method="post" style="display: none;">
      <input type="hidden" id="hidden-file" name="file">
      <input type="hidden" id="hidden-output-file-name" name="output-filename">
    </form>
    {% with messages = get_flashed_messages() %}
      {% if messages %}
        <div class="flash-message">
          {% for message in messages %}
            <p>{{ message }}</p>
          {% endfor %}
        </div>
      {% else %}
        <div class="flash-message">
          <p>File not uploaded yet</p>
        </div>
      {% endif %}
    {% endwith %}
  </div>
  <script>
    function submitProcessForm() {
      const fileInput = document.getElementById('file-upload');
      const outputFileNameInput = document.getElementById('output-file-name');
      const hiddenFileInput = document.getElementById('hidden-file');
      const hiddenOutputFileNameInput = document.getElementById('hidden-output-file-name');

      if (fileInput.files.length > 0 && outputFileNameInput.value) {
        hiddenFileInput.value = fileInput.files[0].name;
        hiddenOutputFileNameInput.value = outputFileNameInput.value;
        document.getElementById('process-form').submit();
      } else {
        alert('Please upload a file and specify an output file name.');
      }
    }
  </script>
</body>
</html>
```

This setup redirects to the /download route after processing the file, ensuring the file is downloaded and the user is redirected back to the index page. Give it a go!

References

1. github.com 2. github.com

You

How can i flip this method so that it replaces only the filename in the path self.input_file (everything after final /) with the filename of the file from file_path?

Copilot

You can achieve this by extracting just the filename from the file_path and then replacing the filename in self.input_file. Here's an updated version of your method:

```
import os

def update_filename_in_path(self, file_path):
    # Get the filename from file_path
    new_filename = os.path.basename(file_path)

    # Get the directory path from self.input_file
    directory_path = os.path.dirname(self.input_file)

    # Create the new path by combining the directory path and the new filename
    new_path = os.path.join(directory_path, new_filename)

    return new_path

# Example usage
class ExampleClass:
    def __init__(self, input_file):
        self.input_file = input_file

    def update_filename_in_path(self, file_path):
        new_filename = os.path.basename(file_path)
        directory_path = os.path.dirname(self.input_file)
        new_path = os.path.join(directory_path, new_filename)
        return new_path

# Assuming your file_path and input_file are like this
example_instance = ExampleClass("https://github.com/HasanBradfordUni/BCFT-
AI/blob/main/Other%20AI%20Projects/AI%20Document%20Extraction/doc/blank/Case%20management%20profile%2
0-%20blank.docx")
file_path = "C:/localFiles/newfile.docx"

updated_path = example_instance.update_filename_in_path(file_path)
print(updated_path)  # Outputs the new path with updated filename
```

This method gets the filename from file_path using os.path.basename and replaces the filename in self.input_file with the new filename while keeping the original directory path intact. This should give you the desired result.

You

I want to do it with regular expressions, like in the method I currently have

Copilot

Sure thing. Here's how you can use regular expressions to replace only the filename in self.input_file with the filename from file_path:

```
import re

class ExampleClass:
    def __init__(self, input_file):
        self.input_file = input_file

    def update_filename_in_path(self, file_path):
        # Regular expression to match everything after the final /
        pattern = r"(.*/)(.*)"

        # Extract the filename from the new file path
        match_new_file = re.match(pattern, file_path)
```

```
        if match_new_file:
            new_filename = match_new_file.group(2)
        else:
            print("No match found for the new file path")
            return self.input_file  # Return the original path if no match found

        # Replace the filename in the original input file path with the new filename
        updated_path = re.sub(pattern, r"\1" + new_filename, self.input_file)

        return updated_path

# Example usage
example_instance = ExampleClass("https://github.com/HasanBradfordUni/BCFT-
AI/blob/main/Other%20AI%20Projects/AI%20Document%20Extraction/doc/blank/Case%20management%20profile%2
0-%20blank.docx")
file_path = "C:/localFiles/newfile.docx"

updated_path = example_instance.update_filename_in_path(file_path)
print(updated_path)  # Outputs the new path with the updated filename
```

In this example:

- The regular expression pattern (.*/)(.*) matches the entire path and groups it into two parts: everything before the last / and everything after (the filename).
- The match_new_file is used to extract the new filename from file_path.
- re.sub is then used to replace the filename in self.input_file with the new filename while keeping the directory part intact.

This approach should give you the desired result using regular expressions.

You

How can I reduce this path, "/workspace/BCFT-AI/Other AI Projects/AI Document Extraction/src/Other AI Projects/AI Document Extraction/src/localFiles/Case%20management%20profile%20-%20blank.docx" to this "/workspace/BCFT-AI/Other AI Projects/AI Document Extraction/src/localFiles/Case%20management%20profile%20-%20blank.docx" in my python script?

Copilot

You can use Python's string manipulation methods to achieve this. Specifically, you can replace the redundant part of the path with an empty string. Here's how you can do it:

```
# Original path
original_path = "/workspace/BCFT-AI/Other AI Projects/AI Document Extraction/src/Other AI Projects/AI Document
Extraction/src/localFiles/Case%20management%20profile%20-%20blank.docx"

# Redundant part to be removed
redundant_part = "/workspace/BCFT-AI/Other AI Projects/AI Document Extraction/src/Other AI Projects/AI Document
Extraction/src/"

# Replace redundant part with an empty string
reduced_path = original_path.replace(redundant_part, "/workspace/BCFT-AI/Other AI Projects/AI Document
Extraction/src/")

print(reduced_path)  # Outputs: /workspace/BCFT-AI/Other AI Projects/AI Document
Extraction/src/localFiles/Case%20management%20profile%20-%20blank.docx
```

In this script:

1. **original_path**: Contains the initial file path.
2. **redundant_part**: The part of the path you want to remove.
3. **replace method**: Replaces the redundant part with the corrected path.

This will give you the desired reduced path. Try it out and see if it works!

You

My issue is that whenever I run if not os.path.exists(filepath) I get teh following error:FileNotFoundError: [Errno 2] No such file or directory: '/workspace/BCFT-AI/Other AI Projects/AI Document Extraction/src/Other AI Projects/AI Document Extraction/src/localFiles/Case%20management%20profile%20-%20blank.docx'My filepath is confirmed to be "Other AI Projects/AI Document Extraction/src/localFiles/Case%20management%20profile%20-%20blank.docx"

Copilot

Got it. The issue seems to be with the file path doubling up, resulting in the path not being found. We need to ensure that the path is corrected before checking if it exists.

Here's a method to clean up the path using regular expressions, ensuring it doesn't contain repeated sections:

```
import re
import os

def clean_up_path(file_path):
    # Regular expression to find repeating parts in the path
    pattern = r"(.*)(\1)"
    cleaned_path = re.sub(pattern, r"\1", file_path)
    return cleaned_path

# Example usage
file_path = "/workspace/BCFT-AI/Other AI Projects/AI Document Extraction/src/Other AI Projects/AI Document Extraction/src/localFiles/Case%20management%20profile%20-%20blank.docx"

# Clean up the path
cleaned_path = clean_up_path(file_path)
print("Cleaned Path:", cleaned_path)  # Should output the corrected path

# Check if the cleaned path exists
if not os.path.exists(cleaned_path):
    print("File not found:", cleaned_path)
else:
    print("File found:", cleaned_path)
```

Explanation:

1. **Pattern**: The regular expression pattern r"(.*)(\1)" finds and removes any repeated parts in the path.
2. **re.sub**: This replaces the repeated part with a single instance of it.
3. **Check existence**: Then, you check if the cleaned path exists using os.path.exists(cleaned_path).

This should help resolve the issue of the incorrect path causing the FileNotFoundError. Try it and let me know if it helps!

You

How does this method work and is there an alternative way to download the file:
return send_file(filepath, as_attachment=True)

Copilot

The send_file method from Flask is used to send files from your server to the client, allowing them to download it. When you use send_file(filepath, as_attachment=True), Flask sends the file at filepath to the browser, prompting a download dialog.

How It Works:

- **send_file Function**: This function serves the file at the specified path.
- **as_attachment=True**: This parameter makes the file download as an attachment. The browser will prompt the user to save or open the file instead of displaying it in-browser.

Here's a quick example of how it's typically used:

```python
from flask import Flask, send_file

app = Flask(__name__)

@app.route('/download')
def download_file():
    filepath = 'path/to/your/file.txt'
    return send_file(filepath, as_attachment=True)

if __name__ == '__main__':
    app.run(debug=True)
```

Alternative Way:

Another approach is to use send_from_directory, which can be useful if you're serving files from a directory:

```python
from flask import Flask, send_from_directory

app = Flask(__name__)

@app.route('/download/<filename>')
def download_file(filename):
    directory = 'path/to/your/directory'
    return send_from_directory(directory, filename, as_attachment=True)

if __name__ == '__main__':
    app.run(debug=True)
```

How to Use:

1. **Place your files** in the specified directory.
2. **Access the route** with the filename you want to download, e.g., /download/yourfile.txt.

This way, you can easily manage and serve multiple files from a directory.

Feel free to try out both methods and see which one works best for your needs. Let me know if you need more details!

You

I'm trying the send_from_directory method, I have this setup:

```python
return redirect(url_for('download_file', filepath=out_file_path))

@app.route('/download')
def download_file():
    filepath = request.args.get('filepath')
    return send_from_directory(folderName, filepath, as_attachment=True)
```

Copilot

Almost there. The send_from_directory function expects the directory and the filename separately. Here's how you can modify your setup:

1. **Extract the directory and filename**:
2. **Modify the download_file function** to use send_from_directory properly.

Here's how you can update your code:

```python
from flask import Flask, request, redirect, url_for, render_template, flash, send_file, send_from_directory
import os

app = Flask(__name__)
app.secret_key = 'supersecretkey'  # Required for flash messages
```

```python
app.config['UPLOAD_FOLDER'] = 'localFiles'
app.config['OUTPUT_FOLDER'] = 'outFolder'
os.makedirs(app.config['UPLOAD_FOLDER'], exist_ok=True)
os.makedirs(app.config['OUTPUT_FOLDER'], exist_ok=True)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/upload', methods=['POST'])
def upload_file():
    if 'file' not in request.files:
        flash('No file part')
        return redirect(url_for('index'))

    file = request.files['file']
    if file.filename == '':
        flash('No selected file')
        return redirect(url_for('index'))

    if file:
        filename = file.filename
        file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        if os.path.exists(file_path):
            flash('File already uploaded')
        else:
            file.save(file_path)
            flash('File successfully uploaded')

    return redirect(url_for('index'))

@app.route('/process', methods=['POST'])
def process_and_download():
    filename = request.form['file']
    output_filename = request.form['output-filename']
    file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)

    if not os.path.exists(file_path):
        flash('File not found')
        return redirect(url_for('index'))

    # Assuming documentExtractor is defined and available
    mapped_data = documentExtractor.extractInfo(file_path)
    out_file_path = os.path.join(app.config['OUTPUT_FOLDER'], output_filename)

    # Save the output file
    documentExtractor.save(out_file_path, mapped_data)

    return redirect(url_for('download_file', filepath=out_file_path))

@app.route('/download')
def download_file():
    filepath = request.args.get('filepath')
    directory = os.path.dirname(filepath)
    filename = os.path.basename(filepath)

    if not os.path.exists(filepath):
        flash('File not found')
        return redirect(url_for('index'))

    return send_from_directory(directory, filename, as_attachment=True)

if __name__ == '__main__':
    app.run(debug=True)
```

Key Changes:

1. **Extract Directory and Filename**:
    o   In the download_file function, directory = os.path.dirname(filepath) extracts the directory path.
    o   filename = os.path.basename(filepath) extracts the filename.
2. **send_from_directory**:
    o   Use send_from_directory(directory, filename, as_attachment=True) to send the file.

Give this a try, and it should fix the issue with downloading the file.

References

1. github.com 2. github.com

I have the following code, however, I just get to a 404 not found url instead of downloading the file, folderName is already defined, teh not found url includes
"download?filepath=https://github.com/HasanBradfordUni/BCFT-AI/blob/main/Other%2520AI%2520Projects/AI%2520Document%2520Extraction/doc/blank/Case%2520man agement%2520profile%2520-%2520blank.docx"

The issue seems to be that the URL in filepath is not correctly formatted for local file paths, and send_from_directory expects a directory and a filename, not a full URL-encoded path.

Here's an updated approach that should solve your problem:

Updated Flask Code

1. **Process and Download Route**:
    o   Ensure the file path is localized correctly.
2. **Download Route**:
    o   Decode the URL and correctly split the folder and file name.

```
from flask import Flask, request, redirect, url_for, render_template, flash, send_file, send_from_directory
import os
import urllib.parse

app = Flask(__name__)
app.secret_key = 'supersecretkey'  # Required for flash messages
app.config['UPLOAD_FOLDER'] = 'localFiles'
app.config['OUTPUT_FOLDER'] = 'outFolder'
os.makedirs(app.config['UPLOAD_FOLDER'], exist_ok=True)
os.makedirs(app.config['OUTPUT_FOLDER'], exist_ok=True)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/upload', methods=['POST'])
def upload_file():
    if 'file' not in request.files:
        flash('No file part')
        return redirect(url_for('index'))

    file = request.files['file']
    if file.filename == '':
        flash('No selected file')
        return redirect(url_for('index'))
```

```
   if file:
       filename = file.filename
       file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
       if os.path.exists(file_path):
           flash('File already uploaded')
       else:
           file.save(file_path)
           flash('File successfully uploaded')

   return redirect(url_for('index'))

@app.route('/process', methods=['POST'])
def process_and_download():
   filename = request.form['file']
   output_filename = request.form['output-filename']
   file_path = os.path.join(folderName, filename)

   if not os.path.exists(file_path):
       flash('File not found')
       return redirect(url_for('index'))

   # Assuming docExt is defined and available
   mapped_data = docExt.extractInfo(file_path)
   docExt.save(output_filename, mapped_data)
   out_file_path = docExt.localiseFilePath(output_filename)

   encoded_path = urllib.parse.quote(out_file_path)
   return redirect(url_for('download_file', filepath=encoded_path))

@app.route('/download')
def download_file():
   encoded_filepath = request.args.get('filepath')
   filepath = urllib.parse.unquote(encoded_filepath)
   directory = os.path.dirname(filepath)
   filename = os.path.basename(filepath)

   if not os.path.exists(filepath):
       flash('File not found')
       return redirect(url_for('index'))

   return send_from_directory(directory, filename, as_attachment=True)

if __name__ == '__main__':
   app.run(debug=True)
```

Explanation

1. **Encoding and Decoding**:
   o When redirecting to the download_file route, the file path is URL-encoded using urllib.parse.quote.
   o In the download_file route, the path is decoded back using urllib.parse.unquote.
2. **Extracting Directory and Filename**:
   o os.path.dirname(filepath) extracts the directory path.
   o os.path.basename(filepath) extracts the filename.

Key Adjustments

- This ensures that URL encoding issues do not interfere with finding the file.
- It helps maintain proper file paths without double slashes or URL-encoded characters causing problems.

Try this setup to see if it resolves the issue.

References

1. github.com 2. github.com

What is an alternate way I can download the file then redirect back to index. If I return a string with the filepath that needs downloading when redirecting to index and then in the html I invoke a javascript function that downloads the file, would that work?

Copilot

That's a solid plan! Here's how you can implement it:

1. **Return the file path in a string** when you redirect to the index page.

2. **Modify your HTML and JavaScript** to trigger a download based on that file path.

Flask Code

In your process_and_download function, return the file path:

```
@app.route('/process', methods=['POST'])
def process_and_download():
    filename = request.form['file']
    output_filename = request.form['output-filename']
    file_path = os.path.join(folderName, filename)

    if not os.path.exists(file_path):
        flash('File not found')
        return redirect(url_for('index'))

    # Assuming docExt is defined and available
    mapped_data = docExt.extractInfo(file_path)
    out_file_path = docExt.localiseFilePath(output_filename)

    # Return the file path as a query parameter when redirecting to index
    return redirect(url_for('index', filepath=out_file_path))
```

HTML Template

Modify your template to include a JavaScript function that triggers the download:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>File Upload</title>
    <style>
        .container {
            width: 400px;
            margin: 0 auto;
            text-align: center;
            padding: 20px;
            border: 1px solid #ddd;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }
        h1 {
            font-size: 2em;
            margin-bottom: 20px;
        }
        label {
            display: block;
            margin: 15px 0 5px;
            font-size: 1.2em;
        }
```

```css
        input[type="file"],
        input[type="text"] {
            width: 100%;
            padding: 10px;
            margin-bottom: 20px;
            border: 1px solid #ccc;
            border-radius: 4px;
        }
        .button-container {
            display: flex;
            justify-content: center;
            gap: 10px;
        }
        button {
            padding: 10px 20px;
            background-color: #4CAF50;
            color: white;
            border: none;
            border-radius: 4px;
            cursor: pointer;
        }
        button:hover {
            background-color: #45a049;
        }
        .flash-message {
            margin-top: 20px;
            padding: 10px;
            border: 1px solid #4CAF50;
            background-color: #d4edda;
            color: #155724;
            border-radius: 4px;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>File Upload</h1>
        <form id="upload-form" action="/upload" method="post" enctype="multipart/form-data">
            <label for="file-upload">File Upload</label>
            <input type="file" id="file-upload" name="file" accept=".doc, .docx">
            <label for="output-file-name">Output File Name</label>
            <input type="text" id="output-file-name" name="output-filename" placeholder="Enter output file name">
            <div class="button-container">
                <button type="submit">Upload</button>
                <button type="button" onclick="submitProcessForm()">Extract Info</button>
            </div>
        </form>
        <form id="process-form" action="/process" method="post" style="display: none;">
            <input type="hidden" id="hidden-file" name="file">
            <input type="hidden" id="hidden-output-file-name" name="output-filename">
        </form>
        {% with messages = get_flashed_messages() %}
            {% if messages %}
                <div class="flash-message">
                    {% for message in messages %}
                        <p>{{ message }}</p>
                    {% endfor %}
                </div>
            {% else %}
                <div class="flash-message">
                    <p>File not uploaded yet</p>
                </div>
            {% endif %}
        {% endwith %}
```

```
    </div>
    <script>
      function submitProcessForm() {
        const fileInput = document.getElementById('file-upload');
        const outputFileNameInput = document.getElementById('output-file-name');
        const hiddenFileInput = document.getElementById('hidden-file');
        const hiddenOutputFileNameInput = document.getElementById('hidden-output-file-name');

        if (fileInput.files.length > 0 && outputFileNameInput.value) {
          hiddenFileInput.value = fileInput.files[0].name;
          hiddenOutputFileNameInput.value = outputFileNameInput.value;
          document.getElementById('process-form').submit();
        } else {
          alert('Please upload a file and specify an output file name.');
        }
      }

      // Function to download the file
      function downloadFile(filePath) {
        const link = document.createElement('a');
        link.href = filePath;
        link.download = filePath.split('/').pop();
        document.body.appendChild(link);
        link.click();
        document.body.removeChild(link);
      }

      // Check if filepath is in the URL
      const urlParams = new URLSearchParams(window.location.search);
      const filepath = urlParams.get('filepath');
      if (filepath) {
        downloadFile(filepath);
      }
    </script>
</body>
</html>
```

Explanation:

1. **Return File Path**: In the process_and_download function, return the file path as a query parameter.
2. **JavaScript Function**: Add a downloadFile function to handle the file download.
3. **Trigger Download**: Check if filepath is present in the URL and call downloadFile if it is.

This setup should trigger the download of the file after processing and then redirect back to the index page. Give it a try!

You

How can I navigate to the directory where teh file is stored ('folderName' in teh python code), then download the file via the javascript in the html, then redirect back to index?

Copilot

Sure thing. Here's how you can achieve this step-by-step:

1. **Modify the Flask app to navigate to the file directory and return the correct path to the HTML**.
2. **Use JavaScript in the HTML to download the file and then redirect back to the index**.

Flask Code:

In your process_and_download function, return the directory and filename to the index page:

```
@app.route('/process', methods=['POST'])
def process_and_download():
    filename = request.form['file']
```

```
        output_filename = request.form['output-filename']
        file_path = os.path.join(folderName, filename)

        if not os.path.exists(file_path):
            flash('File not found')
            return redirect(url_for('index'))

        # Assuming docExt is defined and available
        mapped_data = docExt.extractInfo(file_path)
        out_file_path = docExt.localiseFilePath(output_filename)

        # Return the directory and filename as query parameters
        directory = os.path.dirname(out_file_path)
        filename = os.path.basename(out_file_path)
        return redirect(url_for('index', directory=directory, filename=filename))
```

HTML Template:

Modify your template to include a JavaScript function that triggers the download and then redirects back to the index page:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>File Upload</title>
    <style>
        .container {
            width: 400px;
            margin: 0 auto;
            text-align: center;
            padding: 20px;
            border: 1px solid #ddd;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }
        h1 {
            font-size: 2em;
            margin-bottom: 20px;
        }
        label {
            display: block;
            margin: 15px 0 5px;
            font-size: 1.2em;
        }
        input[type="file"],
        input[type="text"] {
            width: 100%;
            padding: 10px;
            margin-bottom: 20px;
            border: 1px solid #ccc;
            border-radius: 4px;
        }
        .button-container {
            display: flex;
            justify-content: center;
            gap: 10px;
        }
        button {
            padding: 10px 20px;
            background-color: #4CAF50;
```

```
                color: white;
                border: none;
                border-radius: 4px;
                cursor: pointer;
            }
            button:hover {
                background-color: #45a049;
            }
            .flash-message {
                margin-top: 20px;
                padding: 10px;
                border: 1px solid #4CAF50;
                background-color: #d4edda;
                color: #155724;
                border-radius: 4px;
            }
        </style>
    </head>
    <body>
        <div class="container">
            <h1>File Upload</h1>
            <form id="upload-form" action="/upload" method="post" enctype="multipart/form-data">
                <label for="file-upload">File Upload</label>
                <input type="file" id="file-upload" name="file" accept=".doc, .docx">
                <label for="output-file-name">Output File Name</label>
                <input type="text" id="output-file-name" name="output-filename" placeholder="Enter output file name">
                <div class="button-container">
                    <button type="submit">Upload</button>
                    <button type="button" onclick="submitProcessForm()">Extract Info</button>
                </div>
            </form>
            <form id="process-form" action="/process" method="post" style="display: none;">
                <input type="hidden" id="hidden-file" name="file">
                <input type="hidden" id="hidden-output-file-name" name="output-filename">
            </form>
            {% with messages = get_flashed_messages() %}
                {% if messages %}
                    <div class="flash-message">
                        {% for message in messages %}
                            <p>{{ message }}</p>
                        {% endfor %}
                    </div>
                {% else %}
                    <div class="flash-message">
                        <p>File not uploaded yet</p>
                    </div>
                {% endif %}
            {% endwith %}
        </div>
        <script>
            function submitProcessForm() {
                const fileInput = document.getElementById('file-upload');
                const outputFileNameInput = document.getElementById('output-file-name');
                const hiddenFileInput = document.getElementById('hidden-file');
                const hiddenOutputFileNameInput = document.getElementById('hidden-output-file-name');

                if (fileInput.files.length > 0 && outputFileNameInput.value) {
                    hiddenFileInput.value = fileInput.files[0].name;
                    hiddenOutputFileNameInput.value = outputFileNameInput.value;
                    document.getElementById('process-form').submit();
                } else {
                    alert('Please upload a file and specify an output file name.');
```

```
            }
        }

        // Function to download the file and then redirect
        function downloadFile(directory, filename) {
            const link = document.createElement('a');
            link.href = `${directory}/${filename}`;
            link.download = filename;
            document.body.appendChild(link);
            link.click();
            document.body.removeChild(link);
            // Redirect back to index after download
            window.location.href = '/';
        }

        // Check if directory and filename are in the URL
        const urlParams = new URLSearchParams(window.location.search);
        const directory = urlParams.get('directory');
        const filename = urlParams.get('filename');
        if (directory && filename) {
            downloadFile(directory, filename);
        }
    </script>
</body>
</html>
```

Explanation:

1. **Return Directory and Filename**: In the process_and_download function, return the directory and filename as query parameters.
2. **JavaScript Function**: Add a downloadFile function that downloads the file and then redirects back to the index.
3. **Trigger Download**: Check if directory and filename are present in the URL and call downloadFile if they are.

This should download the file from the specified directory and then redirect back to the index page. Give it a shot!

You

Would it work if instead of doing what we're currently doing in the javascript, we set the window.location.href to the directory appended to teh current path first, then we download filename in downloadFile and then redirect back?

Copilot

That approach can work as well. You can navigate to the directory first, trigger the download, and then redirect back. Here's how you can adjust the JavaScript:

Updated JavaScript in HTML Template

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>File Upload</title>
    <style>
        .container {
            width: 400px;
            margin: 0 auto;
            text-align: center;
            padding: 20px;
            border: 1px solid #ddd;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }
        h1 {
            font-size: 2em;
```

```
        margin-bottom: 20px;
    }
    label {
        display: block;
        margin: 15px 0 5px;
        font-size: 1.2em;
    }
    input[type="file"],
    input[type="text"] {
        width: 100%;
        padding: 10px;
        margin-bottom: 20px;
        border: 1px solid #ccc;
        border-radius: 4px;
    }
    .button-container {
        display: flex;
        justify-content: center;
        gap: 10px;
    }
    button {
        padding: 10px 20px;
        background-color: #4CAF50;
        color: white;
        border: none;
        border-radius: 4px;
        cursor: pointer;
    }
    button:hover {
        background-color: #45a049;
    }
    .flash-message {
        margin-top: 20px;
        padding: 10px;
        border: 1px solid #4CAF50;
        background-color: #d4edda;
        color: #155724;
        border-radius: 4px;
    }
  </style>
</head>
<body>
  <div class="container">
    <h1>File Upload</h1>
    <form id="upload-form" action="/upload" method="post" enctype="multipart/form-data">
      <label for="file-upload">File Upload</label>
      <input type="file" id="file-upload" name="file" accept=".doc, .docx">
      <label for="output-file-name">Output File Name</label>
      <input type="text" id="output-file-name" name="output-filename" placeholder="Enter output file name">
      <div class="button-container">
        <button type="submit">Upload</button>
        <button type="button" onclick="submitProcessForm()">Extract Info</button>
      </div>
    </form>
    <form id="process-form" action="/process" method="post" style="display: none;">
      <input type="hidden" id="hidden-file" name="file">
      <input type="hidden" id="hidden-output-file-name" name="output-filename">
    </form>
    {% with messages = get_flashed_messages() %}
      {% if messages %}
        <div class="flash-message">
          {% for message in messages %}
            <p>{{ message }}</p>
          {% endfor %}
```

```
        </div>
    {% else %}
        <div class="flash-message">
            <p>File not uploaded yet</p>
        </div>
    {% endif %}
    {% endwith %}
    </div>
    <script>
        function submitProcessForm() {
            const fileInput = document.getElementById('file-upload');
            const outputFileNameInput = document.getElementById('output-file-name');
            const hiddenFileInput = document.getElementById('hidden-file');
            const hiddenOutputFileNameInput = document.getElementById('hidden-output-file-name');

            if (fileInput.files.length > 0 && outputFileNameInput.value) {
                hiddenFileInput.value = fileInput.files[0].name;
                hiddenOutputFileNameInput.value = outputFileNameInput.value;
                document.getElementById('process-form').submit();
            } else {
                alert('Please upload a file and specify an output file name.');
            }
        }

        function navigateToDirectoryAndDownload(directory, filename) {
            // Set the window location to the directory first
            window.location.href = `${window.location.origin}${directory}`;

            // Wait for the location change then download the file
            setTimeout(() => {
                const link = document.createElement('a');
                link.href = `${directory}/${filename}`;
                link.download = filename;
                document.body.appendChild(link);
                link.click();
                document.body.removeChild(link);

                // Redirect back to index after download
                window.location.href = '/';
            }, 500);  // Adjust the timeout as needed
        }

        // Check if directory and filename are in the URL
        const urlParams = new URLSearchParams(window.location.search);
        const directory = urlParams.get('directory');
        const filename = urlParams.get('filename');
        if (directory && filename) {
            navigateToDirectoryAndDownload(directory, filename);
        }
    </script>
</body>
</html>
```

Explanation:

1. **Navigate to Directory**: The navigateToDirectoryAndDownload function sets window.location.href to navigate to the directory first.
2. **Download File**: After navigating to the directory, it triggers the download of the file.
3. **Redirect Back**: Once the file is downloaded, it redirects back to the index page.

This approach ensures that the browser navigates to the directory first, downloads the file, and then redirects back to the index page. Adjust the timeout as needed to ensure the location change completes before triggering the download. Try it out!

Intel HAXM is required to run this AVD. Android Emulator is incompatible with Hyper-V. Unfortunately, you cannot have Hyper-V running and use the emulator. Here is what you can do: 1) Start a command prompt as Administrator 2) Run the following command: C:\Windows\system32> bcdedit /set hypervisorlaunchtype off 3) Reboot your machine.
I keep getting the above message, I've followed the instructions and rebooted 2 or 3 times but it keeps happening

Copilot

It sounds like you're encountering a common issue with Hyper-V and Intel HAXM. Here are a few steps you can try to resolve this:

1. **Disable Hyper-V**:
   - Open Command Prompt as Administrator and run:
     bcdedit /set hypervisorlaunchtype off
   - Reboot your machine.
2. **Turn off Windows Defender's Core Isolation**:
   - Go to **Settings** > **Update & Security** > **Windows Security** > **Device Security**.
   - Click on **Core isolation details** and turn off **Memory integrity**.
   - Reboot your machine.
3. **Check BIOS Settings**:
   - Ensure that Intel Virtualization Technology (VT-x) is enabled in your BIOS/UEFI settings.
4. **Verify Hyper-V Status**:
   - Open Command Prompt as Administrator and run:
     bcdedit /enum
   - Check if Hyper-V is listed as off.
5. **Reinstall HAXM**:
   - If the issue persists, try uninstalling and reinstalling Intel HAXM.

If you've already tried these steps and the problem continues, there might be another underlying issue. Could you let me know if you're using any specific virtualization software or if there are any error messages in the Command Prompt when you run the bcdedit command?

References

github.com

You

(Now Tuesday, Nov 5)
I have the following Python code but I'm not getting teh expected output, the output that I want is below:
Code:

```
#SQL statement generator
import re

data_table = []
temp_statement = input("Enter a template SQL statement: ")
data = " "
while data != "":
  data = input("Enter a data row (Enter to finish): ")
  data_row = data.split()
  data_table.append(data_row)

if temp_statement.startswith("INSERT"):
  start_index = temp_statement.rfind("VALUES (")
  for num in range(int(temp_statement.count(",")/2)):
    pattern = r'.{' + str(start_index) + r'}([^,]*)'
    match = re.search(pattern, temp_statement)
```

```
    temp_statement = temp_statement.replace(match.group(1), data_table[0][num])
    start_index = temp_statement.rfind(match.group(1))
  print(temp_statement)
else:
  print("Support for this type of SQL statement is coming soon!")
```

Actual Output:

Enter a template SQL statement: INSERT INTO `actor`(`actor_id`, `first_name`, `last_name`) VALUES
('[value-1]','[value-2]','[value-3]')
Enter a data row (Enter to finish): 201 HARRY CON
Enter a data row (Enter to finish): 202 MARGERET BOOKER
Enter a data row (Enter to finish): 204 HOLLY MATON
Enter a data row (Enter to finish): 206 AMAAN QURAISHI
Enter a data row (Enter to finish):

Traceback (most recent call last):
File "C:\Users\fifau\OneDrive - University of Bradford\Documents\Python_Files\SQl statement gen.py", line
17, in <module>
temp_statement = temp_statement.replace(match.group(1), data_table[0][num]) ^
                                        ~~~~~~~~~~~

AttributeError: 'NoneType' object has no attribute 'group'

Expected Output:

Enter a template SQL statement: INSERT INTO `actor`(`actor_id`, `first_name`, `last_name`) VALUES ('[value-
1]','[value-2]','[value-3]')
Enter a data row (Enter to finish): 201 HARRY CON
Enter a data row (Enter to finish): 202 MARGERET BOOKER
Enter a data row (Enter to finish): 204 HOLLY MATON
Enter a data row (Enter to finish): 206 AMAAN QURAISHI
Enter a data row (Enter to finish):

SQL Statement is as follows:
INSERT INTO `actor`(`actor_id`, `first_name`, `last_name`) VALUES (201,'HARRY','CON'),
(202,'MARGERET','BOOKER'), (204,'HOLLY','MATON'), (206,'AMAAN','QURAISHI')

Copilot

I see where the problem lies. The issue is with how the pattern is constructed and used to find and replace placeholders in the SQL template. Additionally, the data_table is collecting empty rows which are causing the error.

Let's revise the code to address these issues and generate the expected output:

Revised Code

```python
import re

# Function to generate SQL statement
def generate_sql_statement(temp_statement, data_table):
    # Check if the SQL statement is of type INSERT
    if temp_statement.startswith("INSERT"):
        # Find the start index of the VALUES section
        start_index = temp_statement.rfind("VALUES (") + len("VALUES (")

        # Extract the placeholders from the template
        pattern = re.compile(r'\[value-(\d+)\]')
        placeholders = pattern.findall(temp_statement)

        # Create the SQL statements for each row in the data table
        sql_rows = []
        for row in data_table:
            values = ','.join([f"'{row[int(ph)-1]}'" for ph in placeholders])
            sql_rows.append(f"({values})")
```

```
    # Combine the SQL statements into one
    result_sql = temp_statement[:start_index] + '), '.join(sql_rows) + ')'
    print(result_sql)
  else:
    print("Support for this type of SQL statement is coming soon!")

# Get the template SQL statement from the user
temp_statement = input("Enter a template SQL statement: ")

# Get the data rows from the user
data_table = []
while True:
  data = input("Enter a data row (Enter to finish): ")
  if not data:
    break
  data_row = data.split()
  data_table.append(data_row)

# Generate the SQL statement
generate_sql_statement(temp_statement, data_table)
```

Explanation:

1. **Pattern and Placeholders**:
   - The regular expression r'\[value-(\d+)\]' is used to find placeholders in the template SQL statement.
2. **Generating SQL Rows**:
   - For each row in the data_table, the placeholders are replaced with the corresponding values.
   - The values are formatted as '{value}' for proper SQL syntax.
3. **Combining SQL Statements**:
   - The generated SQL rows are combined into one statement with the VALUES clause.

This should produce the expected output where the placeholders in the SQL template are replaced with the provided data rows.

Try running this revised code and see if it gives you the desired results! If you encounter any issues, feel free to share the details.