Collab-2-Panopto--UoW by the University of Westminster, London, UK

Daniel Bidemi and Fabienne Bonzon, BSD, University of Westminster, Oct 2022

Introduction

The University of Westminster was interested in further adapting and developing the *Collab-2-Panopto* application, which migrates video recordings between two different student services, Blackboard Collaborate and Panopto, to support the practice of hybrid learning cost-effectively.

This application, *Collab-2-Panopto--UoW*, is mainly built from the work of Lee Bardon and Josh Bruylant *Collab-2-Panopto* and several other python scripts *(see credits),* then further adapted by the University of Westminster. The application gives students the opportunity to have their course videos always available and reduces the burden placed on Blackboard Collaborate due to its increase in course related content and recordings during the Covid-19 period and beyond.

Main Additions

1. New functionality added to read Blackboard Collaborate information from a **database**. This functionality was implemented due to the overload on the Collab API. We witnessed that the high number of results returned from the Collab API caused performance issues due to the number of API calls being made and the number of courses being unnecessarily queried on the API daily, which then consequently increases the load on the cloud service. Therefore, reading Collab data from a database lead to significantly better performance management.

2. We built **error handling** to mitigate cloud service performance issues around the main application integration points:
   - Database
   - BB Collab
   - Panopto

   Further standard application-level error handling and notification functionality was introduced to support code maintenance and improve application flow, as well as to mitigate application integration points fatal errors.

3. Implemented a ***globalConfig.py*** *(found in application config folder)* – Allows flexibility around main application integration points actions such as configurable *BB Collab delete* action, *Panopto time sleep* to mitigate against lag when discovering sessions, and preferred *data source option DB or INLINE.*

Collab to Panopto Workflow

1. Finds all recordings added to database within a period specified by the user.
2. Finds corresponding course folders on Panopto. If recording has no corresponding folder, it will be mapped to a default Panopto folder (found in config.py).
3. Downloads Collab recording(s) to local server
4. Uploads recording(s) to corresponding Panopto folder
5. Where uploads are successful:
   - Renames recording on Panopto
   - Deletes recording from Collaborate
   - Deletes recording from local server
6. Where uploads are not successful:
   - Deletes recording from Panopto
   - Deletes from local server
7. Where uploads are not discovered:
   - Mitigation step to re-discover
     1. Renames recording on Panopto
     2. Deletes recording from Collaborate
     3. Deletes from local server

Environment Set-Up

- Clone the project to the your desired server
- Install Python 3.9
- Obtain the most appropriate version of the package manager Miniconda for Python 3.9 [https://docs.conda.io/en/latest/miniconda.html]
- Ensure that Python package manager 'Pip' [https://pip.pypa.io/en/stable/installing/] is installed on the server
- Download ODBC driver from: *https://learn.microsoft.com/en-us/sql/connect/odbc/download-odbc-driver-for-sql-server?view=sql-server-ver16*

Once the above steps have been completed, navigate to the root directory of the project and run:

```
conda env create -f environment.yml
```

This will create a virtual environment and download all packages used by the application (and their dependencies) into it. Should any errors occur during this step, conda will provide instructive error messages that should enable the problem to be solved. Dependencies can also be examined manually by viewing the environment.yml file.

On successful completion of the above, activate the project's environment by running:

```
conda activate collaborate-to-panopto
```

On some occasions, you would have to enter 'activate base' in the prompt before you can activate your collaborate-to-panopto virtual environment.

Once your virtual environment has been created and activated, run:

```
pip install pid

pip install psutil

pip install pyodbc
```

API, SSL & Download folder Configuration

The application interacts with two external REST API services: namely the Blackboard Collaborate REST API, and the Panopto REST API. It also requires configuration details to enable successful sending of alert emails to service owners and maintainers.

To enable this process, a configuration template *ConfigTemplate.py* is provided in the */config/* folder. Please copy this into a new file, fill out the relevant details, and save the file as Config.py within the same directory.

A *downloads* folder is also needed to store the recordings temporarily when they have been downloaded from the Blackboard Collaborate API. Please create a 'downloads' folder at the root directory of your cloned copy of the project.

Config.py

Collaborate Settings

Your University will have its own Collaborate key and secret details.

Panopto Settings

For development purposes, it is instructive to create a client_id and client_secret_id via the Panopto website:

1. Sign in to Panopto website
2. Click your name in right-upper corner, and click "User Settings"
3. Select "API Clients" tab
4. Click "Create new API Client" button
5. Enter arbitrary Client Name
6. Select User Based Server Application type.
7. Enter http://localhost/ into CORS Origin URL.
8. Enter http://localhost:9127/redirect into Redirect URL.
9. The rest can be blank. Click "Create API Client" button.
10. Note the created Client ID and Client Secret. "ppto_server": "your.university.server" "ppto_client_id": obtained client_id "ppto_client_secret": obtained client_secret

Database settings

Your university will have its own database details. Fill in your university database details into the config file:

```
…
db_connection = {
   "Driver": "ODBC Driver",
   "server" : "serverName",
   "database" : "databaseName",
   "username":"databaseUserName",
   "password":"databaseUserPassword",
   "TrustServerCertificate":"yes",
   "Encrypt":"yes",
   "Trusted_Connection": "yes",
}
…
```

To Run

Once the environment has been successfully set-up and activated, and *Config.py* has been created and updated with the appropriate credentials, the program can be launched.

The *runscript.py* file is located in the project root directory. Here, we can set the period over which the automatic pathway searches for recordings listed in the database. The period is measured in days, and the default period is 1 day. This is adjusted via the argument of the function call set_search_start_date() from the *Utilities.py* module.

Launch

To launch, run the following from the root directory:

```
python runscript.py
```

This will print-- PRESS ENTER FOR MANUAL RUN --and launch a countdown timer (default 5 seconds). If you do not press enter during this 5-second period, the automatic pathway will be launched. Otherwise, the manual pathway will be launched. The duration of the countdown timer can be adjusted via the **main** function in *runscript.py*.

When manual run has been selected, two options are available: Enter 'a' for actual run or 'p' for preview. Entering ‘a’ will start the application and allow you to manually select the dates and migrate recordings from the specified dates. Entering ‘p’ will start preview mode, detailed below. Any other input will alert user to try again, entering either ‘a’ or ‘p’.

Preview Mode

Preview mode allows launching a "dry run" where the application will simulate what it would do, without actually doing anything. This is useful when looking for information on what launching the application would do if started.

Using this mode will display the below information:

- **Courses on Collab**: lists all courses added to Collaborate database between search start-date and present
- **Recordings per course**: lists all recordings found for the courses listed above
- **Corresponding folders on Panopto**: lists all folders found on Panopto that correspond to the courses found on Collaborate. Courses with no folder on Panopto will be mapped to a default Panopto folder
- **Planned uploads**: lists all recordings that can be uploaded, and their destination folder on Panopto

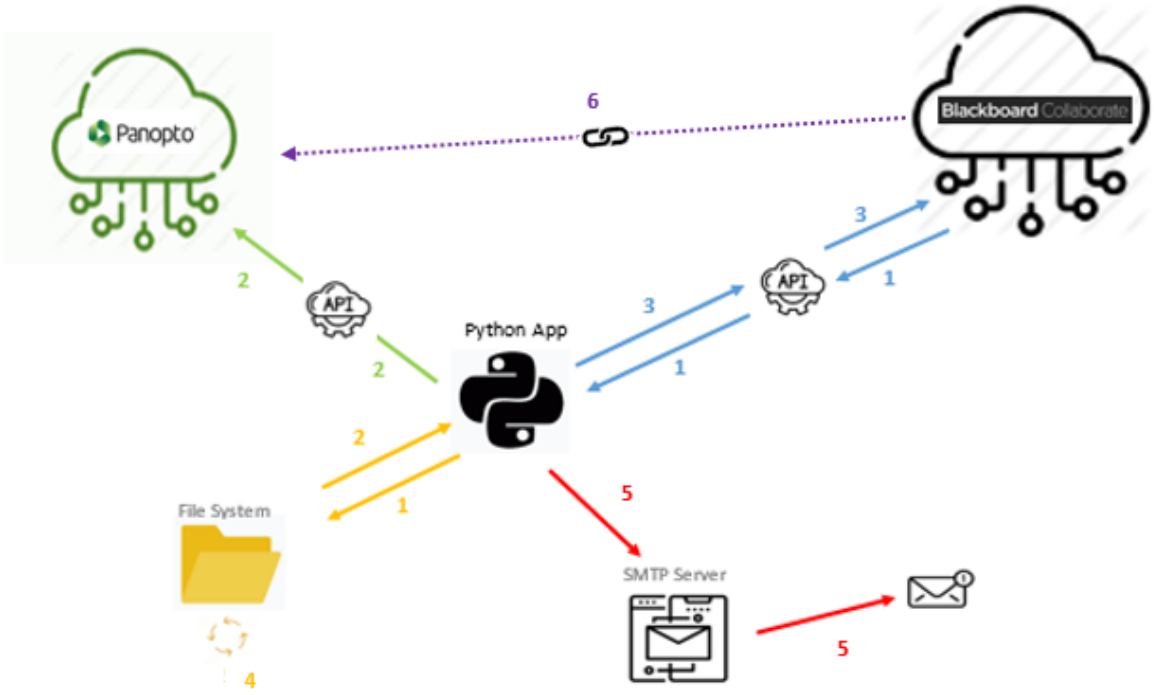
PID File locking mechanism

Upon launching the application, a pid file is created in */home/user/tmp/runscript.py.pid*. This stops two applications running at the same time, which would cause problems.

As long as this file exists, no two instances of the application can run at the same time.

If the application encounters an unexpected crash or is killed ungracefully then this pid file will remain in place and stop any further executions. This is useful to realise an exception has happened and may be worth investigating. The pid file is safe to delete when the problem has been investigated and steps have been taken to repair what the sudden stop might have caused.

High-Level Application Flow



Automated nightly - move videos from BB Collaborate to Panopto

1 - **Download** videos from BB Collaborate course area via API call and save to temporary file storage

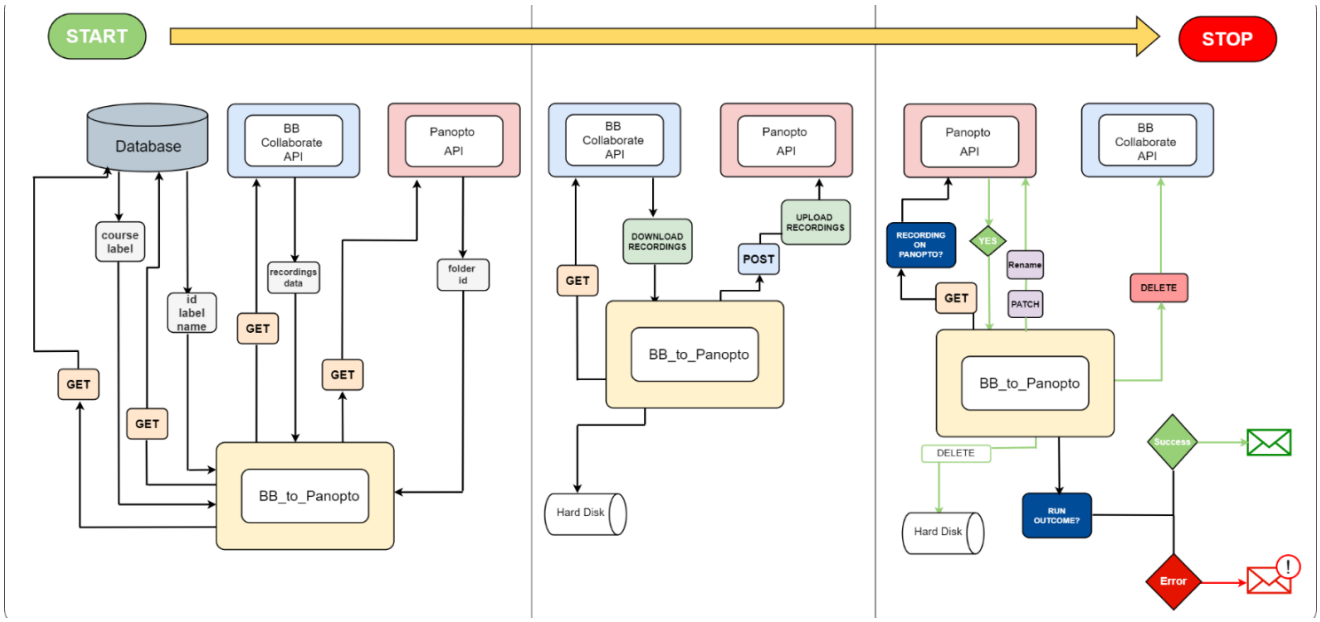
2 - **Upload** videos to Panopto corresponding course folder via API call from temporary file storage

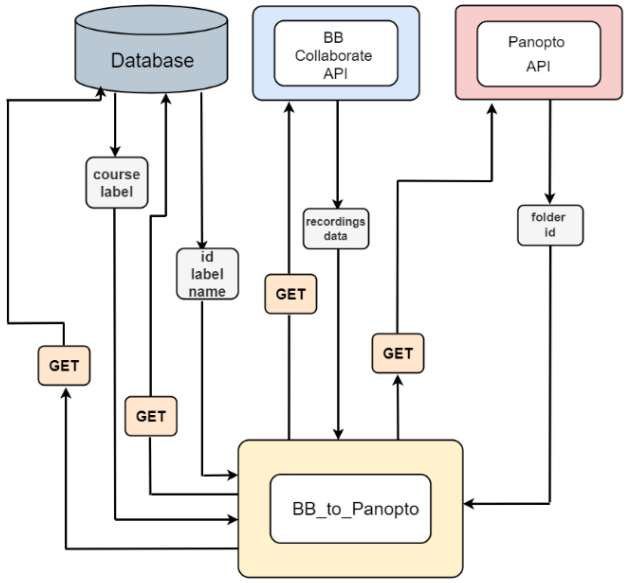
3 - **Delete** videos from BB Collaborate via API call

4 - **Delete** local copies of videos

5 - **Email** notifications

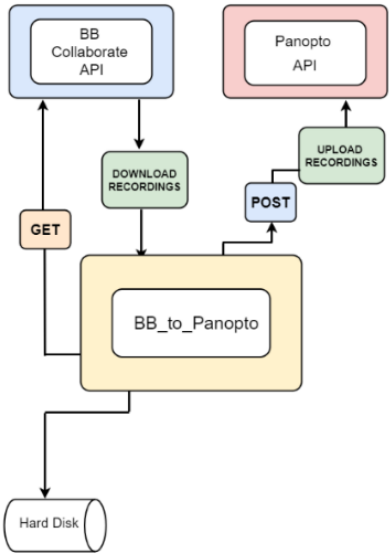
6 - Manual remedial action following mapping failure (Panopto folders not found)
Some Panopto folders may have been renamed or not even created when the course was manually set up initially. If this is the case, the folder name will have to be edited back to its original name, the missing link in Collab created and the videos manually migrated to Panopto.

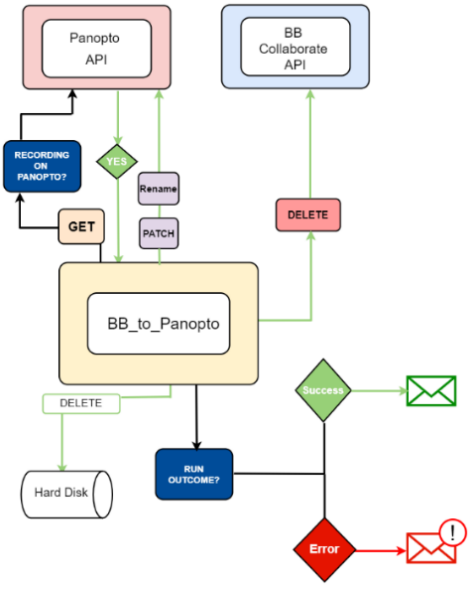


<table>
<tr>
<td> 
</td>
<td>The application will query the database to return course labels of given time period (set in runscript.py)

It will query the database with given course labels, to return the course name, label, and the list of recording ids corresponding with each course label.

It will use the information returned to retrieve the recording data (recording object/s) from the BB-Collaborate API.

It will use the course label and course name to retrieve their corresponding folder id on Panopto using the Panopto API.</td>
</tr>
<tr>
<td> 
</td>
<td>The application will download recordings from BB collaborate API and save them to the local server downloads folder.

Once recordings are downloaded in the local downloads folder, the app will upload these recordings to Panopto.</td>
</tr>
<tr>
<td> 
</td>
<td>Once the app has discovered the recordings on Panopto, it will:
1. rename recordings on Panopto
2. delete recordings on Blackboard
3. delete recordings on local server

Run outcome
<u>Successful run</u>
Email alert with information on run summary

<u>Un-successful run</u>
Email alert notifying that an error has occurred. Email will come with attachments (traceback, debug log, info log) to help with investigation.</td>
</tr>
</table>

References

Authors

- Daniel Bidemi - *DanielAlao*
- Fabienne Bonzon -*bonzonf*

License

This project is licensed under the MIT License - see the LICENSE.md file for details

Credits

This project was inspired by and adapted from:

Collab-2-Panopto: https://github.com/leebardon/Collab-2-Panopto - MIT License

PyCollab: https://github.com/zerausolrac/PyCollab - MIT License

OSCELOT: https://github.com/OSCELOT/Collab-Panopto - Blackboard Open Source License

BulkExportBBCollaborateRecordings:
https://github.com/SimonXIX/BulkExportBBCollaborateRecordings - MIT License