**Video processing**

Step 1: Extraction of videos’ clips from YouTube

Technology used: Laravel (PHP) and Python

Library used: YouTube-dl (YouTube-dl is a Command-line program to download videos from YouTube.com and other video sites)

Graphical user interface, application

Description automatically generated

*Result is shown in interface as flash message and contains following:*

*Success = All successful processed id*

*Failed = All videos unavailable id*

*id\_why\_failed= id of all videos with unknown reason for failure*

*SC= Success count  
FC= Failed count*

*IDF=I Don’t know why failed count*

*Time for the process is in seconds*

Time required for processing (internet speed and quality of video impacts the time):

500 videos = 3 hrs.

1450 videos = 7 hrs.

63 videos = 20 minutes

4 videos = 1.5 minutes

For single video it just returns result as success upon successful extraction:

A picture containing graphical user interface

Description automatically generated

Though the entire software structure consists of various part two main part of php program and python program is as below:

NOTE: Program and software structure can be improved drastically, no architecture or standard is followed (except web interface uses MVC architecture), it is just bunch of “if-else” statements used to obtain desired result, program can be made more modular and understandable, however since its single person with not so much time and also focus is not to develop industry standard software no attention was paid towards software design.

PHP

    {

        $starttime = microtime(true);

        set\_time\_limit(36000);

*if* ($request->multiple != null) {

            $result            = explode(",", $request->multiple);

            $id\_collection     = [];

            $failed\_collection = [];

            $idk\_why\_fail=[];

*foreach* ($result as $id) {

                $process = new Process(['python', 'C:\Users\Invictus\PycharmProjects\videoprocessing\main.py', trim($id)]);

                $process->setTimeout(36000);

                $process->run();

*if* (!$process->isSuccessful()) {

                    array\_push($idk\_why\_fail, trim($id));

*continue*;

                }

                $output = $process->getOutput();

                $result = preg\_split('/\r\n|\r|\n/', $output);

*if* ($result[1] == "Failed") {

                    array\_push($failed\_collection, trim($id));} *else* {

                    array\_push($id\_collection, trim($id));

                }

            }

            $success\_count = 0;

            $failure\_count = 0;

            $idk\_why\_fail\_count = 0;

            $success\_id    = "Success : ";

*foreach* ($id\_collection as $id) {

                $success\_id = $success\_id . $id . " , ";

                $success\_count++;

            }

            $success\_id = $success\_id . "Failed : ";

*foreach* ($failed\_collection as $id) {

                $success\_id = $success\_id . $id . ",";

                $failure\_count++;

            }

            $success\_id = $success\_id . "id\_why\_Failed : ";

*foreach* ($idk\_why\_fail as $id) {

                $success\_id = $success\_id . $id . " , ";

                $idk\_why\_fail\_count++;

            }

            $success\_id = $success\_id . " SC : " . $success\_count . ", FC : " . $failure\_count.", IDF : " . $idk\_why\_fail\_count;

*/\* do stuff here \*/*

        $endtime  = microtime(true);

        $timediff = $endtime - $starttime;

        $success\_id=$success\_id." Time : ".round($timediff,1)."s";

*return* back()->with('success', $success\_id);

        } *else* {

            $id      = $request->single;

            $process = new Process(['python', 'C:\Users\Invictus\PycharmProjects\videoprocessing\main.py', $id]);

            $process->run();

*if* (!$process->isSuccessful()) {

*throw* new ProcessFailedException($process);

            }

*return* back()->with('success', "Success");

        }

    }

Python:

*# all required library imported*

*import* cv2

*import* numpy *as* np

*import* youtube\_dl

*import* sys

*import* math

*import* os

*import* re

*from* random *import* randint

*if* \_\_name\_\_ == '\_\_main\_\_':

    video\_id = "To7jJ\_d\_kbM"

*# video\_id = sys.argv[1]*

    video\_url = "To7jJ\_d\_kbM"

*# video\_url = sys.argv[1]*

    url = "https://www.youtube.com/watch?v=" + video\_id

    directory = r'C:/xampp/htdocs/processed\_image\_python'

    ydl\_opts = {}

*# create youtube-dl object*

    ydl = youtube\_dl.YoutubeDL(ydl\_opts)

*# set video url, extract video information*

*try*:

        info\_dict = ydl.extract\_info(video\_url, download=False)

*except*:

        print("Failed")

        sys.exit()

*# get video formats available*

    formats = info\_dict.get('formats', None)

    length = math.floor(info\_dict['duration'])

    title = info\_dict["title"]

    regex = re.compile('[^a-zA-Z0-9()]')

    title = regex.sub('\_', title)

*# print(formats)*

*for* f *in* formats:

*# print(f.get('format\_note', None))*

*if* f.get('format\_note', None) == '144p':

            url = f.get('url', None)

*break*

*if* f.get('format\_note', None) == '360p':

            url = f.get('url', None)

*break*

*if* f.get('format\_note', None) == '240p':

            url = f.get('url', None)

*break*

    cap = cv2.VideoCapture(url)

*# check if url was opened*

*if* not cap.isOpened():

        print('video not opened')

        exit(-1)

    frame\_rate = math.floor(cap.get(5))

*# print(length, frame\_rate)*

    os.chdir(directory)

*if* not (os.path.isdir(directory + "/" + title + "-" + video\_id)):

        os.mkdir(title + "-" + video\_id)

    os.chdir(directory + "/" + title + "-" + video\_id)

    flag = 0

*if* length > 28740:

        length = 28740

    gap = math.floor(length / 8)

*# print(gap)*

*while* True:

*# read frame*

        ret, frame = cap.read()

*if* flag == 0 or 1 or 2 or 3 or 4 or 5 or 6 or 7:

            point = ((flag \* gap) + gap)

            cap.set(1, point \* frame\_rate)

*if* flag == 8:

*break*

*# check if frame is empty*

*if* ret:

*# cv2.imshow('frame', frame)*

            minute = point // 60

            print(minute, length, point)

*if* length < 60:

*if* flag == 0:

                    file\_name = "-" + str(point) + "s.png"

*else*:

                    print(file\_name)

                    cv2.imwrite(f"clip" + file\_name, frame)

                    file\_name = "-" + str(point) + "s.png"

*if* cv2.waitKey(30) & 0xFF == ord('q'):

*break*

*if* length > 60 and 60 > minute:

                minute = point // 60

                print(flag)

                second = point - (minute \* 60)

*if* flag == 0:

                    file\_name = "-" + str(minute) + "m-" + str(second) + "s.png"

*else*:

                    print(file\_name)

                    cv2.imwrite(f"clip" + file\_name, frame)

                    file\_name = "-" + str(minute) + "m-" + str(second) + "s.png"

*if* cv2.waitKey(30) & 0xFF == ord('q'):

*break*

*if* minute >= 60:

*# print(file\_name)*

                hr = minute // 60

                minute = (point - (hr \* 60 \* 60)) // 60

                second = point - ((minute \* 60) + (hr \* 60 \* 60))

                cv2.imwrite(f"clip" + file\_name, frame)

                file\_name = "-" + str(hr) + "h-" + str(minute) + "m-" + str(second) + "s.png"

*if* cv2.waitKey(30) & 0xFF == ord('q'):

*break*

*else*:

*break*

        flag += 1

    cap.release()

    print("Success")

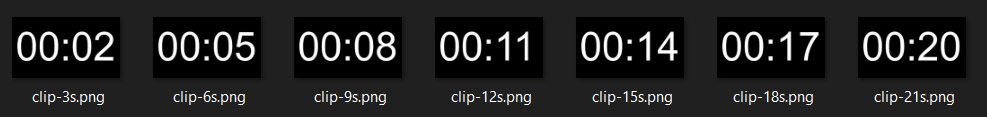
cv2.destroyAllWindows()

OUT OF 2028 VIDEOS: 1900 sucessfully processed, 115 videos unavailable for various reasons(channel removed or video removed or violated youtube guideline, etc), 12 videos output unknown.  
12 videos may have been repeated or something, we can find out if it was repeated but for now if the video is already processed I have just overwritten previous file with new output, if we need confirmation of what happened with those 12, and may be more when we process 17000 videos, we can resolve the issue accordingly.

Some test samples are as follow:

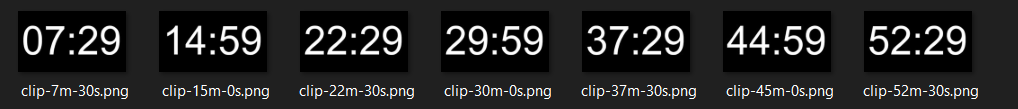
Testing on 30 s length video( video length 32 second):

url: <https://www.youtube.com/watch?v=ot7aXVMtE_g>



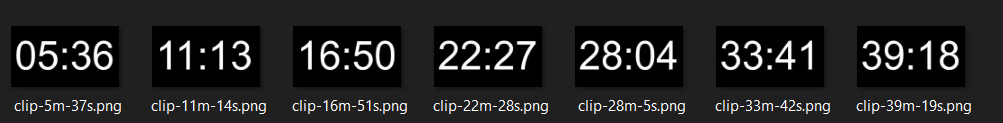
Testing on 1h length video(video length 1 hr 1 second):

url: <https://www.youtube.com/watch?v=vdqcge_SPnc&t=5s>



Testing on 45mins length video (video length 45mins 1second):

url: <https://www.youtube.com/watch?v=Hfs-oNEiEf0&t=50s>



Drawbacks:  
If video is above length of 8hrs, gap between clips may not be even and 7 clips on interval of each hour is extracted. Can be resolved with few lines of code but since I found only one video among 2000 videos to be above 8hrs, at the moment the case is as it is.

Testing on 10hrs length video (video length 10hrs 9second):

url: <https://www.youtube.com/watch?v=CyIhcyCOCkk>

Graphical user interface, text

Description automatically generated

List of unavailable videos:

| MTOv6StMBfg | 4JkgmQIH7PM | xanfG5Rx2cU | lkM3omotHrw | sS0BF5709sA | rqyL4p-RznQ | yigurOl5Iuw | bQ\_pjt0nwuE | i3RIeekJEZo | NqdbOnJFNDA | MhMpIfa2o\_0 | Yqk-VJmIOzc | URRaFK48c70 | VrSSIS45\_n4 | tEnrIS8Vnm4 | MTYduB6Ru-Q | MVfDJfz2bCo | RewfjV8oMps | ed62wnXcIeI | hIykqE8DQNo | z6vs0EAPPbQ | \_mNPMx8Ofos | 3Z32zfRdFVM | 5fkz4jLj9Yw | h4v66YoW\_nM | 95r6MANBl6k | UQrqA-3eUgk | Lg84120HHi8 | aOockEyQb0g | SD-eO-nxeOw | Ldht0W6gE8Q | p26SScjre8o | w1WalIOFx3Y | WtMHJdzrSk8 | -hy4wpkbHOI | KemAh8vVm4g | ttArYumMlns | rMkRpfjCqkc | CIoQkIqdQW0 | S4Rq76KxMi0 | o8ujMT5F6RM | m-ERmE\_ydz4 | RsfzmsUjjKc | 7kNcssyd9sM | hwOPCS98T8A | F4y6l\_rEqDY | nNrS2WollZo | EvkKXfxTMLg | fHepi2Jc9NI | lK5ZhlJ1CA4 | zIgSNSc2Lts | 6\_IjYfJAit4 | 2lFqF4jNJzk | UUZ-UbjbXvc | hhbXDUAKRTY | kuib8EfNf2s | J5\_SGgqHGwE | ggqyc-SJRZM | cl03NPNUD5A | 9XiEnvVLJ9k | ph\_vyHrG69k | EHH23yF65Ic | N7aG2BgIJwg | MbKVtdkdigs | EtIJNbvLzZI | 6J-dlIB0F1M | Ni2d1DpNNRY | L5oh43bZSnU | gbxr7Pw1e\_8 | n9b3lGcS8PE | sHb5KRSTlgs | ZBTzNMgQjc8 | PAsp559Zglo | uaAOoYvO5dU | tnkfcxHKQsc | 2rr86xfkEB4 | m0Qy2spPnnA | \_yh3Fy7zjNg | 06bzv1Vw7TU | IJaV\_Fk3xSY | -ZwPbMkEFV8 | OwHFbX3QFLM | EtWskvBmHBo | yxhaaqE8-0k | N4fNK-gm6IY | JmLCjNfFoCo | p5PJV\_z5P20 | m5x9UmRXnG8 | RQ8QUsX53zs | UWXy9PTmgkM | bbEuJo\_VLCE | fT7e75KxGaw | EvbXbaK3P2U | 8HHQ7rPfzjE | xuEw2J\_4rxM | UniEWbqhoIY | vWUW9qjFzrM | rp3kK4DNu3o | P1ENqSS-i2I | 9uZ2jh7VYSE | gpVwT9tHWYw | pOWmWvQ4ff0 | p-YCjBBCMcc | MqCQmjSGXnc | id9CzIfKlaU | rytZZa1OLxA | \_cARoyGHFPQ | gpzlXQOdkYg | vipMF4wRlrE | o8Act9ZvRZQ | H8nTITJxrCc | aMAxk9pNA8I | yw9\_ufUZ0s0 | 3xqEByeVsbM | EahiV8y3pBY |