Open-Source Technology Use Report

Proof of knowing your stuff in CSE312

Guidelines

Provided below is a template you must use to write your report for each of the technologies you use in your project.

Here are some things to note when working on your report, specifically about the **General Information & Licensing** section for each technology.

- **Code Repository**: Please link the code and not the documentation. If you'd like to refer to the documentation in the **Magic** section, you're more than welcome to, but we'd like to see the code you're referring to as well.
- **License Type**: Three letter acronym is fine.
- **License Description**: No need for the entire license here, just what separates it from the rest.
- **License Restrictions**: What can you *not* do as a result of using this technology in your project? Some licenses prevent you from using the project for commercial use, for example.
- **Who worked with this?**: It's not necessary for the entire team to work with every technology used, but we'd like to know who worked with what.

Also, feel free to extend the cell of any section if you feel you need more room.

If there's anything we can clarify, please don't hesitate to reach out! You can reach us using the methods outlined on the course website or see us during our office hours.

PyCloudinary

General Information & Licensing

Code Repository	https://github.com/cloudinary/pycloudinary
License Type	MIT License
License Description	● use, copy, modify, merge, distribute, publish, sublicense, and sell copies of the software
License Restrictions	● Virtually none
Who worked with this?	Zaki

Use as many of the sections below as needed, or create more, to explain every function, method, class, or object type you used from this library/framework.

cloudinary.config()

Purpose

- What does this tech do for you in your project?
    - The config function must be called before an image can be uploaded to my cloudinary account. The parameters I send are my Cloudinary Account's Cloud Name, API key, and API secret
- This function is used in line 16 of postHanders.py in the function uploadImage

Magic ⋆★⁺₊˚⋆˚☽˚⁀➷｡˚★彡⋆༄

- Config is defined in this file: https://github.com/cloudinary/pycloudinary/blob/a61a9687c8933f23574c38e27f201358e540ee64/cloudinary/__init__.py in line 220
- It takes the parameter keywords and updates the global _config variable using its update() method.

- The update method is defined in this file: https://github.com/cloudinary/pycloudinary/blob/a61a9687c8933f23574c38e27f201358e540ee64/cloudinary/__init__.py#L171 in line 171. Update is a part of the BaseConfig class. The BaseConfig class has a dictionary named __dict__. Update adds a key value pair to __dict__. For our implementation, the keys are 'CLOUD_NAME', 'API_KEY', and 'API_SECRET', which we send along with their respective values in postHandlers.py. Once these key-value pairs are set, we can reference them using _config.api_key, _config.api_secret, and _config.cloud_name.

cloudinary.uploader.upload()

Purpose

Replace this text with some that answers the following questions for the above tech:

- The uploader module contains the upload function, which uploads a file
- We use upload() in line 19 of postHandlers.py. It is given an image that is represented in bytes.

Magic ⋆★｡˚⋆.˚ ☽ ˚｡⋆⁀➴｡˚★彡⋆ ༄

- The upload function is defined in this file: https://github.com/cloudinary/pycloudinary/blob/master/cloudinary/uploader.py in line 48. The parameters it takes are file and options. We do not send any options so I will not discuss this parametrer.

- The upload file returns call_cacheable_api(‘upload’, params, file=file, **option).
    - Params and options are not equal to anything in our implementation, we only send the image bytes to cloudinary.uploader.upload()

- Call_cacheable_api is defined in this file: https://github.com/cloudinary/pycloudinary/blob/a61a9687c8933f23574c38e27f201358e540ee64/cloudinary/uploader.py#L430 in line 430.
    - The parameters it receives are action, params, http_headers, return_error, unsigned, file, timeout, and options
        - The only parameters that our implementation sends this function are action = ‘upload’ and file= the bytes of the image
    - This function returns the variable result, which is set equal to the return value of the function call_api

- Call_api is defined in this file: https://github.com/cloudinary/pycloudinary/blob/a61a9687c8933f23574c38e27f201358e540ee64/cloudinary/uploader.py#L444 in line 444.
    - The parameters it receives are action, params, http_headers, return_error, unsigned, file, timeout, and **options
        - The parameters of our concern are the same as call_cacheable_api; action = ‘upload’ and file = the bytes of the image
    - The function gets an oath_token from cloudinary.config, which we have described above. We have not set an oath_token so the expression on line 456 will evaluate to true. This will execute utils.sign_request() and set the return equal to params
    - Once the parameters are set, the function generates an api_url by calling utils.cloudinary_api_url(action, **options), where action=’upload’ and **options = null.
    - The function will execute the if-elif-else block from lines 469 to 494. The else statement at line 491 will evaluate to true, because file is a bytearray.
    - Once data is set, it will be appended to the parameter list
    - In line 504, the function will set the variable response = the result of a POST request to the cloudinary api. It will send my Cloudinary name, secret, key, and the bytes of the image I want to upload in param_list.
    - Response is then decoded, and, result is set equal to this object in line 511. Result contains a key named ‘url’, which has a value that is equal to the link to the image which is now hosted on Cloudinary. This function then returns result.

- Utils.sign_request is defined in this file: https://github.com/cloudinary/pycloudinary/blob/a61a9687c8933f23574c38e27f201358e540ee64/cloudinary/utils.py#L568 in line 568.
    - This function gets the api key and the api secret (which we already provided in lines 16-18 in postHandlers.py) by referencing cloudinary.config().api_key and cloudinary.config().api_secret, and returns it in a list along with a signature.