Lead IQ assignment – Documentation – Imgur Image Upload API

Contents:

1. Requirements :

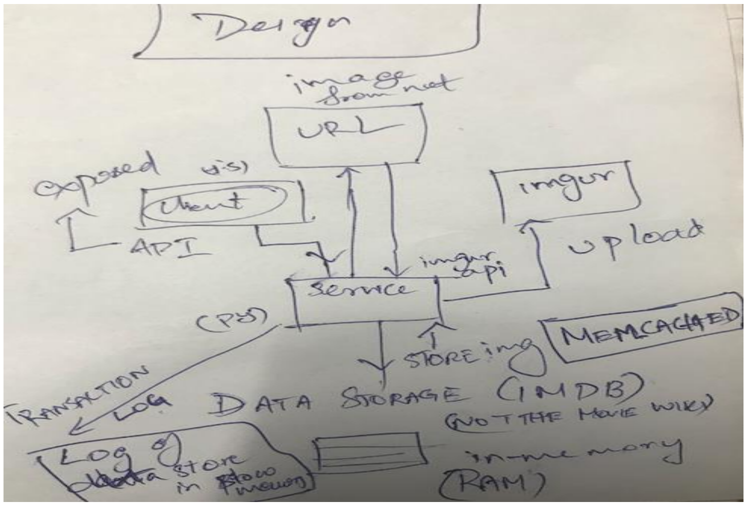
To design a system, that exposes API's to upload jobs where each jobs contain URL for images that need to be downloaded and uploaded to IMGUR site. Each URL will have corresponding imgur url.

2. Language: Python3.7 | Packages: Turbo REST API , IMGUR API, URL

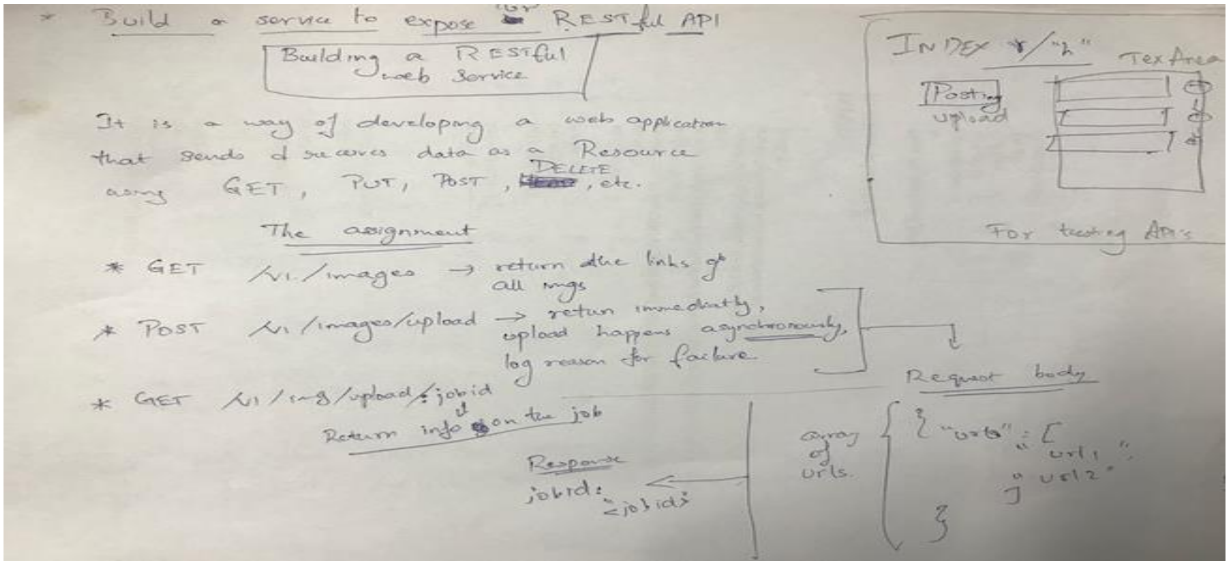
3. System Design:

1. Design Architecture (informal designs):

Initial Over view



Primary Requirements


* Build a service to expose for RESTful API

Building a RESTful web Service

It is a way of developing a web application that sends & receives data as a Resource using GET, PUT, POST, DELETE, etc.

The assignment

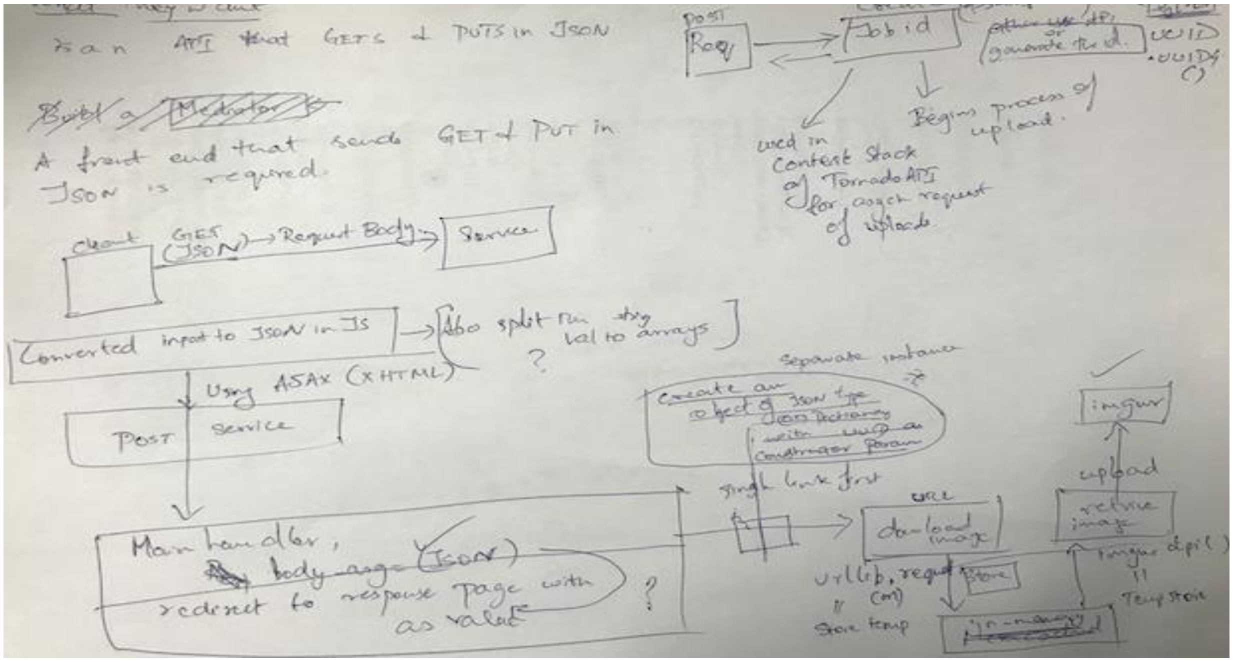
* GET /v1/images → return the links of all imgs
* POST /v1/images/upload → return immediately, upload happens asynchronously, log reason for failure
* GET /v1/img/upload/:jobid
Return info on the job

Response
jobid: <jobid>

Request body
array of urls.
```
{ "urls": [
    "url1",
    "url2"
  ]
}
```

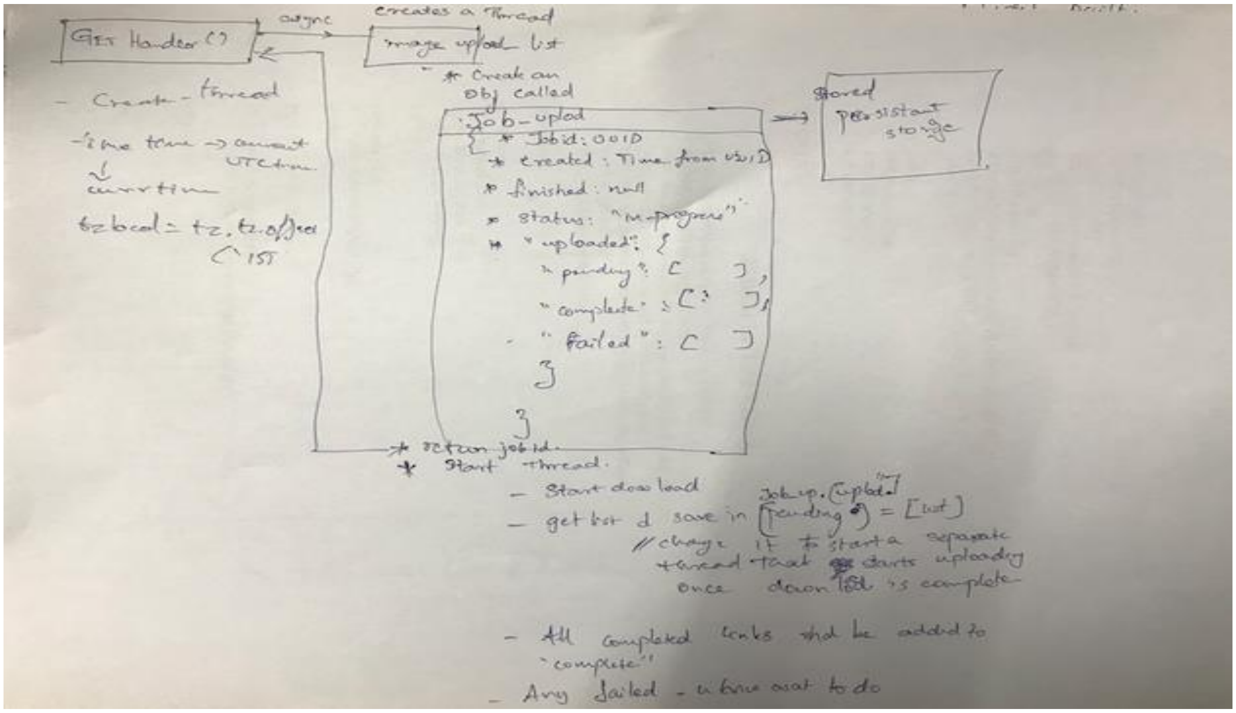
INDEX "/"
Posting upload
TexArea
For testing API's

Control Flow Design



is an API that GETs & PUTs in JSON
~~Build a Mediator~~
A front end that sends GET & PUT in JSON is required.
POST
Req
Jobid
guarantee the id.
UUID
.UUID4()
used in Context Stack of Tornado API for async request of uploads
Begins process of upload.
Client
GET (JSON)
Request Body
Service
Converted input to JSON in JS
Also split the string val to arrays
?
Using AJAX (XHTML)
POST
Service
Main handler,
body args (JSON)
redirect to response page with as value
?
Separate instance
Create an object of JSON type
single link first
URL
download image
urllib, request (or) store temp
Store
in-memory
retrieve image
upload
imgur
Temp store

Data Model For JobID



GET Handler ()
async
creates a Thread
image upload list
* Create an obj called
Job_upload
* Jobid: UUID
* created: Time from UUID
* finished: null
* status: "in-progress"
* "uploaded": {
"pending": [ ],
"complete": [ ],
"failed": [ ]
}
}
Stored persistant storage
- Create - thread
- time time → convert UTCtime
currtime
tzlocal = tz.tz.offset
IST
* return jobid.
* Start thread.
- Start download
- get list & save in Job_up.(upload)(pending) = [list]
// change it to start a separate thread that starts uploading once download is complete
- All completed links shd be added to "complete"
- Any failed - a bin what to do

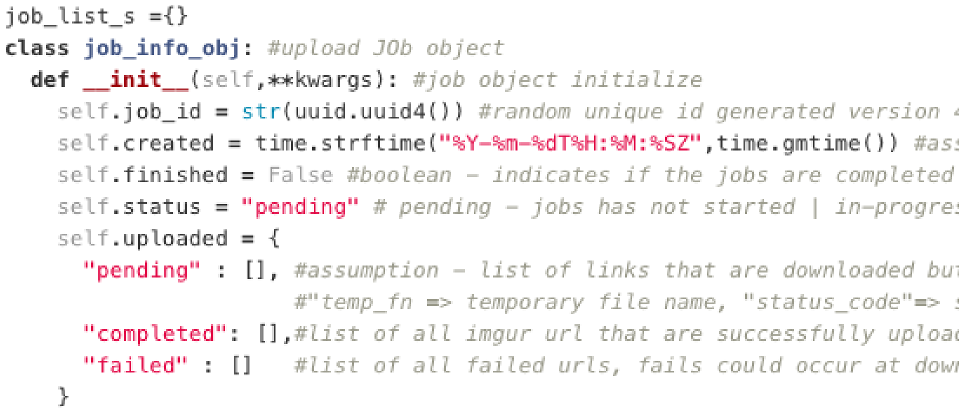
2. API's

The two primary API's:

a. POST /v1/images/upload

The request body contains the list of URL's to download the images and pass it to the Imgur API to be uploaded.

- Post method is handled by the **UploadHandler**() in **main.py**.
- The URL list extracted from the Request body is passed on to the **ImageLoad** module.
- At the ImageLoad module, the queue image URL creates a **Job object** that represents the upload Job along with a unique id using the UUID library. It uses the UUID4 version which is a random id generated that is unique for each session.



```
job_list_s ={}
class job_info_obj: #upload JOb object
  def __init__(self,**kwargs): #job object initialize
    self.job_id = str(uuid.uuid4()) #random unique id generated version
    self.created = time.strftime("%Y-%m-%dT%H:%M:%SZ",time.gmtime()) #as
    self.finished = False #boolean - indicates if the jobs are completed
    self.status = "pending" # pending - jobs has not started | in-progre
    self.uploaded = {
      "pending" : [], #assumption - list of links that are downloaded bu
                      #"temp_fn => temporary file name, "status_code"=>
      "completed": [],#list of all imgur url that are successfully uploa
      "failed" : []   #list of all failed urls, fails could occur at dow
    }
```

- Each job object is stored in a Job list dictionary
- The **begin_upload** Thread is then invoked and the **jobID** is returned to the POST function which is given returned as the Response body.
- Within the begin_upload thread, each url is first downloaded using URLLIB library function and stored in a temp_file, whose path, request status and original link values are updated as a report dictionary format.
- If the download was a success, then the report dictionary is added to the **PENDING** list of the JOB_object.uploaded member.
- Else the url is added to the **FAILED** list of the JOB_object.uploaded member.
- Each of the list in the Pending value is queued for upload and passed to the **uploadImage()** of the **imgurapi.py** module
- Using **upload_from_path()** api method that IMGUR api provide, the images are uploaded.
- If success, the response in the form of URL is store in the JOB_object.uploaded's **SUCCESS**
- If failed, the original URL is appended to the **FAILED** list.
- Once all the PENDING list is exhausted, the status of the job is changed to **complete** and the **finished** boolean is set to true.

b. **GET /v1/images/upload: jobID**

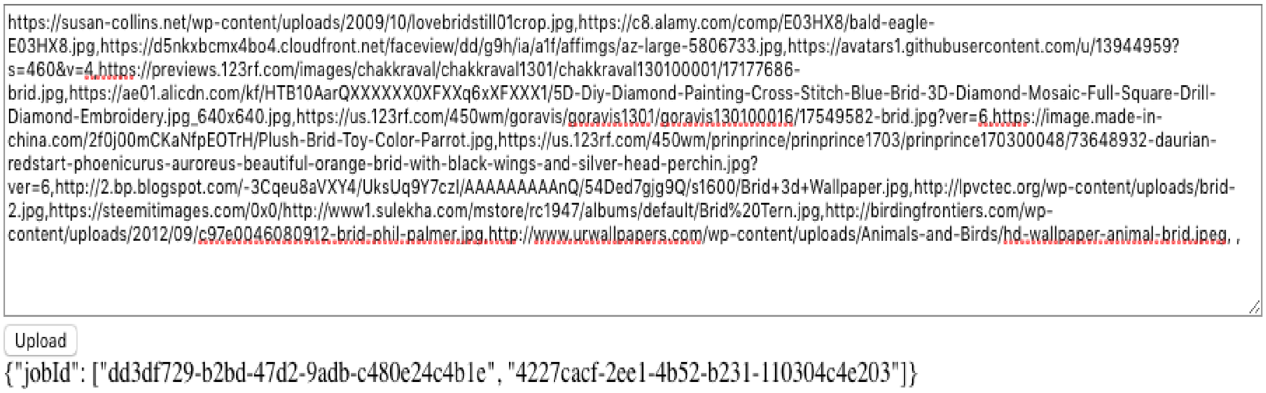
- The GET method with JobID as query string is handled by **JobStatusHandler** in main.py
- The jobId, which was the value from the previous response body, is used as key value to search for the job in the **job_list_s[jobId]** dictionary member of the ImageLoad module.
- If the jobId exists the Get method will return a JSON format of the JOB_object with information on its current status.
- If the jobId does not exist, 505 error is thrown.
- Job information is available even during the download and upload phase.

c. **GET /v1/images**

- The GET method with the above API request is handled by **UploadLinks** handler.
- This api does not have any parameters, therefore it is assumed that the request for list of links that are uploaded to IMGUR, are all the links that have been uploaded in all the jobs regardless of the Jobs status or its completion.
- All the key value pairs in the **job_list_s** dictionary is looped through to get the list of all imgur URL in the **SUCCESS** list of the uploaded list for each JOB.
- Returns a JSON formated list of URL's.

4.USE-CASES:

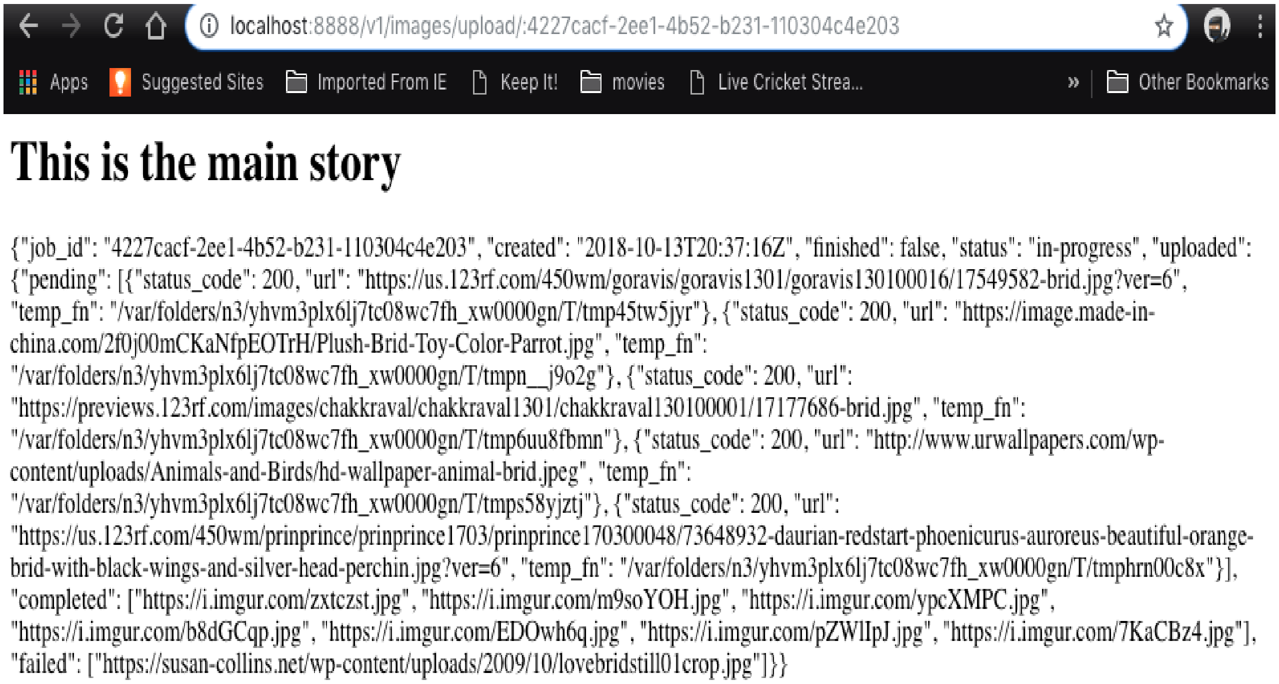
a. **Inputs with variable list size, has duplicates and empty values: POST /v1/images/upload**



```
https://susan-collins.net/wp-content/uploads/2009/10/lovebridstill01crop.jpg,https://c8.alamy.com/comp/E03HX8/bald-eagle-
E03HX8.jpg,https://d5nkxbcmx4bo4.cloudfront.net/faceview/dd/g9h/ia/a1f/affimgs/az-large-5806733.jpg,https://avatars1.githubusercontent.com/u/13944959?
s=460&v=4,https://previews.123rf.com/images/chakkraval/chakkraval1301/chakkraval130100001/17177686-
brid.jpg,https://ae01.alicdn.com/kf/HTB10AarQXXXXXX0XFXXq6xXFXXX1/5D-Diy-Diamond-Painting-Cross-Stitch-Blue-Brid-3D-Diamond-Mosaic-Full-Square-Drill-
Diamond-Embroidery.jpg_640x640.jpg,https://us.123rf.com/450wm/goravis/goravis1301/goravis130100016/17549582-brid.jpg?ver=6,https://image.made-in-
china.com/2f0j00mCKaNfpEOTrH/Plush-Brid-Toy-Color-Parrot.jpg,https://us.123rf.com/450wm/prinprince/prinprince1703/prinprince170300048/73648932-daurian-
redstart-phoenicurus-auroreus-beautiful-orange-brid-with-black-wings-and-silver-head-perchin.jpg?
ver=6,http://2.bp.blogspot.com/-3Cqeu8aVXY4/UksUq9Y7czI/AAAAAAAAAnQ/54Ded7gjg9Q/s1600/Brid+3d+Wallpaper.jpg,http://lpvctec.org/wp-content/uploads/brid-
2.jpg,https://steemitimages.com/0x0/http://www1.sulekha.com/mstore/rc1947/albums/default/Brid%20Tern.jpg,http://birdingfrontiers.com/wp-
content/uploads/2012/09/c97e0046080912-brid-phil-palmer.jpg,http://www.urwallpapers.com/wp-content/uploads/Animals-and-Birds/hd-wallpaper-animal-brid.jpeg, ,
```

```
{"jobId": ["dd3df729-b2bd-47d2-9adb-c480e24c4b1e", "4227cacf-2ee1-4b52-b231-110304c4e203"]}
```

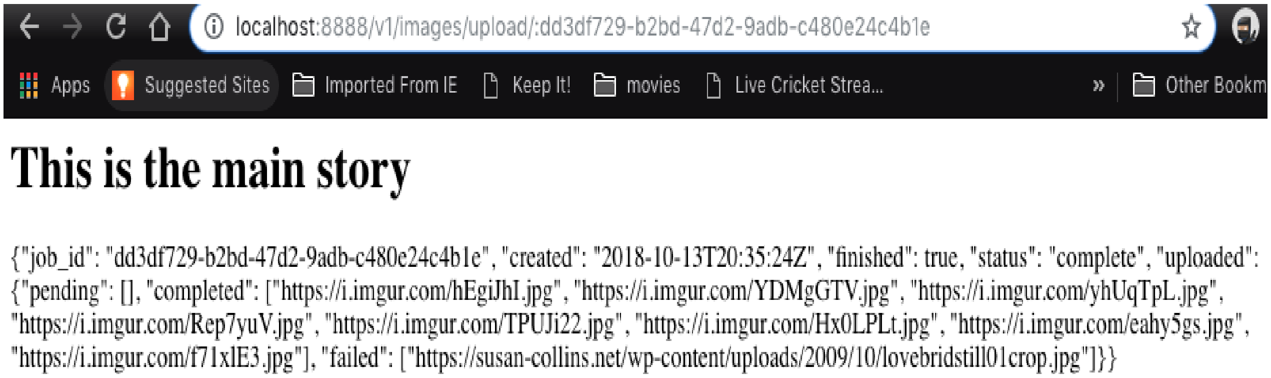
b. **Get status of a Job that is in progess: GET /v1/images/upload/: jobId**



This is the main story

```
{"job_id": "4227cacf-2ee1-4b52-b231-110304c4e203", "created": "2018-10-13T20:37:16Z", "finished": false, "status": "in-progress", "uploaded":
{"pending": [{"status_code": 200, "url": "https://us.123rf.com/450wm/goravis/goravis1301/goravis130100016/17549582-brid.jpg?ver=6",
"temp_fn": "/var/folders/n3/yhvm3plx6lj7tc08wc7fh_xw0000gn/T/tmp45tw5jyr"}, {"status_code": 200, "url": "https://image.made-in-
china.com/2f0j00mCKaNfpEOTrH/Plush-Brid-Toy-Color-Parrot.jpg", "temp_fn":
"/var/folders/n3/yhvm3plx6lj7tc08wc7fh_xw0000gn/T/tmpn__j9o2g"}, {"status_code": 200, "url":
"https://previews.123rf.com/images/chakkraval/chakkraval1301/chakkraval130100001/17177686-brid.jpg", "temp_fn":
"/var/folders/n3/yhvm3plx6lj7tc08wc7fh_xw0000gn/T/tmp6uu8fbmn"}, {"status_code": 200, "url": "http://www.urwallpapers.com/wp-
content/uploads/Animals-and-Birds/hd-wallpaper-animal-brid.jpeg", "temp_fn":
"/var/folders/n3/yhvm3plx6lj7tc08wc7fh_xw0000gn/T/tmps58yjztj"}, {"status_code": 200, "url":
"https://us.123rf.com/450wm/prinprince/prinprince1703/prinprince170300048/73648932-daurian-redstart-phoenicurus-auroreus-beautiful-orange-
brid-with-black-wings-and-silver-head-perchin.jpg?ver=6", "temp_fn": "/var/folders/n3/yhvm3plx6lj7tc08wc7fh_xw0000gn/T/tmphrn00c8x"}],
"completed": ["https://i.imgur.com/zxtczst.jpg", "https://i.imgur.com/m9soYOH.jpg", "https://i.imgur.com/ypcXMPC.jpg",
"https://i.imgur.com/b8dGCqp.jpg", "https://i.imgur.com/EDOwh6q.jpg", "https://i.imgur.com/pZWlIpJ.jpg", "https://i.imgur.com/7KaCBz4.jpg"],
"failed": ["https://susan-collins.net/wp-content/uploads/2009/10/lovebridstill01crop.jpg"]}}
```

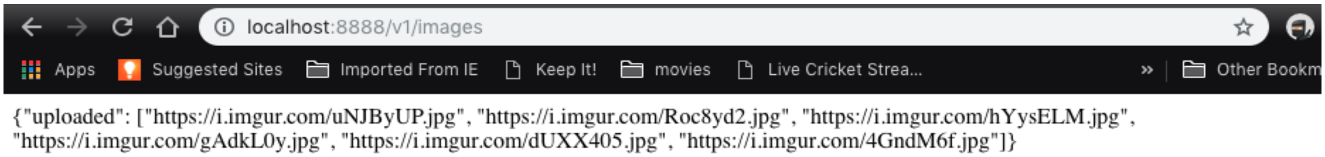
c. **Get status of a Job that has completed: GET /v1/images/upload/: jobId**



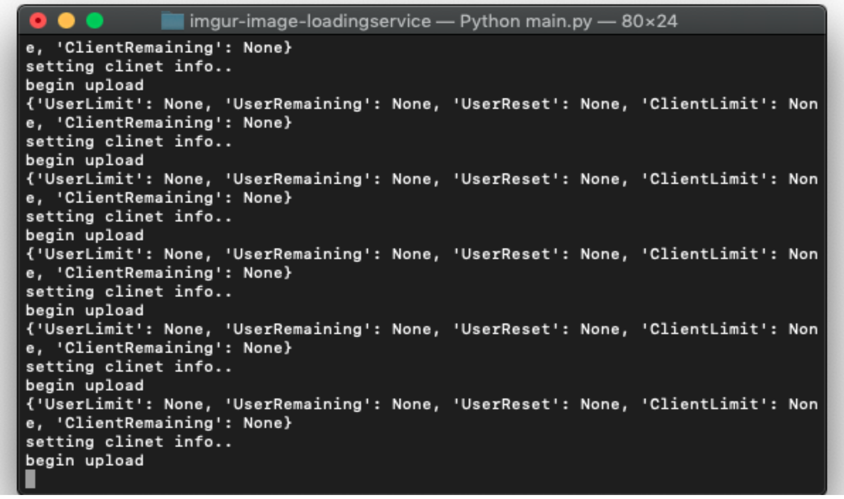
This is the main story

```
{"job_id": "dd3df729-b2bd-47d2-9adb-c480e24c4b1e", "created": "2018-10-13T20:35:24Z", "finished": true, "status": "complete", "uploaded":
{"pending": [], "completed": ["https://i.imgur.com/hEgiJhI.jpg", "https://i.imgur.com/YDMgGTV.jpg", "https://i.imgur.com/yhUqTpL.jpg",
"https://i.imgur.com/Rep7yuV.jpg", "https://i.imgur.com/TPUJi22.jpg", "https://i.imgur.com/Hx0LPLt.jpg", "https://i.imgur.com/eahy5gs.jpg",
"https://i.imgur.com/f71xlE3.jpg"], "failed": ["https://susan-collins.net/wp-content/uploads/2009/10/lovebridstill01crop.jpg"]}}
```

d. **Get list of all uploaded images, the list is available even if the other urls in the Pending is being uploaded: GET /v1/images**



```
{"uploaded": ["https://i.imgur.com/uNJByUP.jpg", "https://i.imgur.com/Roc8yd2.jpg", "https://i.imgur.com/hYysELM.jpg",
"https://i.imgur.com/gAdkL0y.jpg", "https://i.imgur.com/dUXX405.jpg", "https://i.imgur.com/4GndM6f.jpg"]}
```



```
e, 'ClientRemaining': None}
setting clinet info..
begin upload
{'UserLimit': None, 'UserRemaining': None, 'UserReset': None, 'ClientLimit': Non
e, 'ClientRemaining': None}
setting clinet info..
begin upload
{'UserLimit': None, 'UserRemaining': None, 'UserReset': None, 'ClientLimit': Non
e, 'ClientRemaining': None}
setting clinet info..
begin upload
{'UserLimit': None, 'UserRemaining': None, 'UserReset': None, 'ClientLimit': Non
e, 'ClientRemaining': None}
setting clinet info..
begin upload
{'UserLimit': None, 'UserRemaining': None, 'UserReset': None, 'ClientLimit': Non
e, 'ClientRemaining': None}
setting clinet info..
begin upload
{'UserLimit': None, 'UserRemaining': None, 'UserReset': None, 'ClientLimit': Non
e, 'ClientRemaining': None}
setting clinet info..
begin upload
█
```

5.READ ME – Instructions for running:

- **Project is available to clone at:** https://gitlab.com/gauthamsuriyam/imgur-image-loadingservice.git
- **At the project folder run the following code to install packages:**
  - python3 -m pip install -r requirements.txt
  - **Note: if the URL downloads fail due to SSL certificate errors, the following instruction will install the certificates, a known error when running the app on PYTHON 3**
  - /Applications/Python\ 3.7/./Install\ Certificates.command
- **Finally execute:** python3 main.py