



CGG Image Validation Application

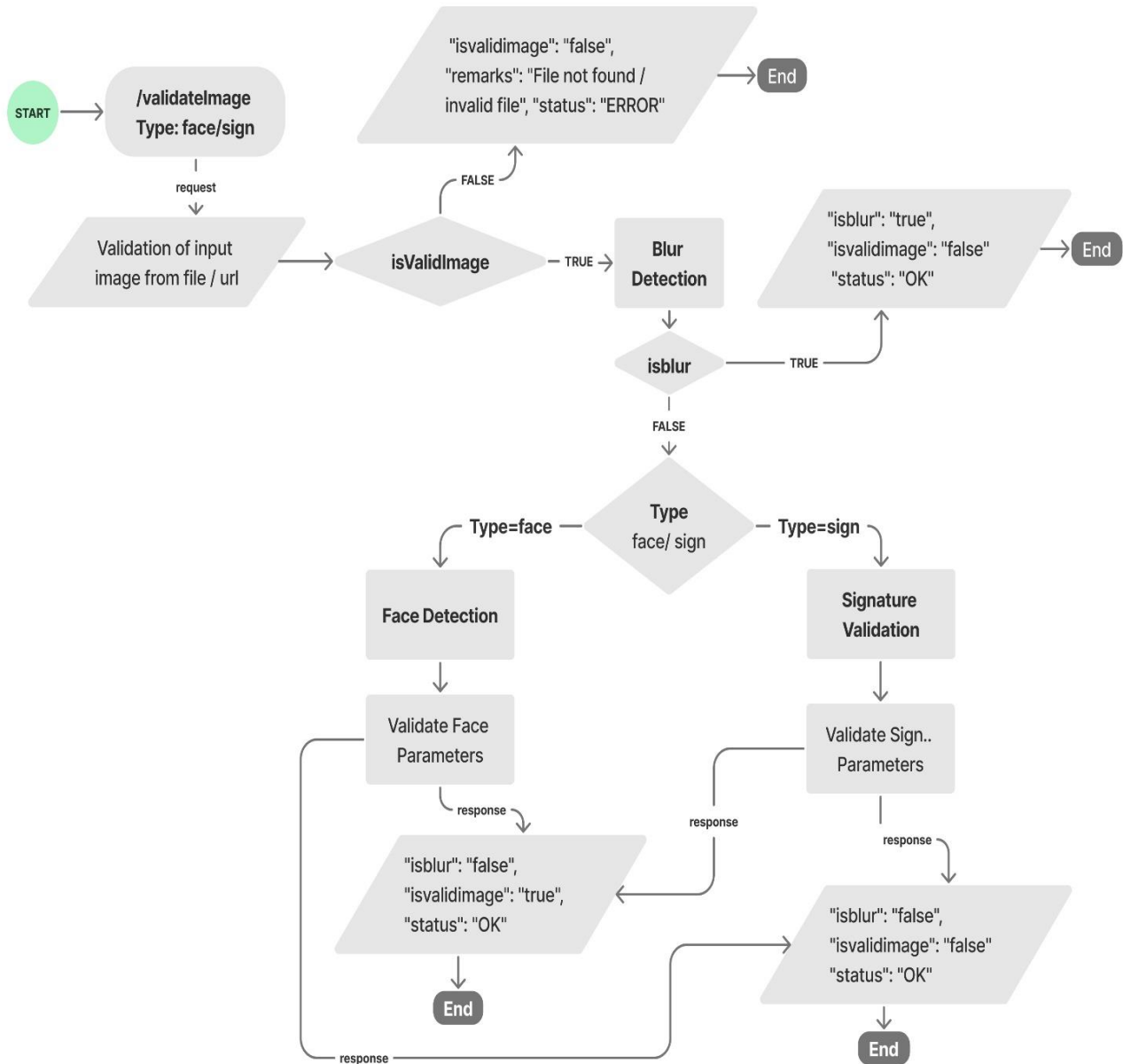
centre for good governance



CGG Image Validation API application setup and usage

Problem Statement: Image validation api implementation for blur detection, face validation, image background verification, signature validation.

Technical Details



Blur Detection:

A function is said to be a piecewise continuous function if it has a finite number of breaks and it does not blow up to infinity anywhere. Let us assume that the function $f(t)$ is a piecewise continuous function, then $f(t)$ is defined using the Laplace transform. The Laplace transform of a function is represented by $L\{f(t)\}$ or $F(s)$. Laplace transform helps to solve the differential equations, where it reduces the differential equation into an algebraic problem. It is the integral transform of the given derivative function with real variable t to convert into a complex function with variable s . For $t \geq 0$, let $f(t)$ be given and assume the function satisfies certain conditions to be stated later on.

Using open-cv we can calculate Laplace transform as mentioned below.

```
# Blur detection with OpenCV  
cv2.Laplacian(image, cv2.CV_64F).var()
```

If the focus measure of the image falls below certain threshold, then image is considered as blurry.



Fig 1. Blurry



Fig 2. Non-Blurry

Face Detection:

Face detection is a process of identifying human faces in images or videos. It is a rapidly expanding area of computer vision that offers a variety of useful applications, such as security systems, face identification, and picture analysis. It is a very used part in Deep Learning. We use face detection for different tasks like login on applications, person recognition and different attendance systems. As Deep learning requires a lot of datasets for training a model, so we can use pretrained models for such type of tasks. This module is built using dlib's state-of-the-art face recognition built with deep learning. The model has an accuracy of 99.38%.

```
# Find all the faces in the image using the default HOG-based model.
```

```
face_locations = face_recognition.face_locations(image)
```

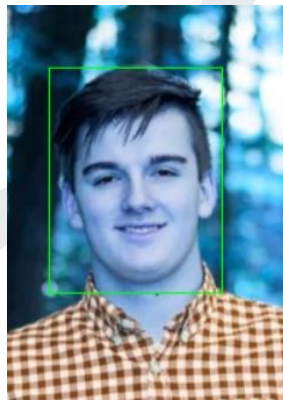


Fig 1. Face Detection

Face Parameters Validation:

The photograph should be in the size of 2-inch x 2-inch or (51 mm x 51 mm) and the resolution of the file should be minimum 350 pixels (Width) X 350 pixels (Height) and maximum 1000 pixels (Width) X 1000 pixels (Height). Bit Depth of image file should be 24 bit and DPI range should be between 200 and 300.

Reference: https://cgifrankfurt.gov.in/public_files/assets/pdf/frankfur10102021.pdf

Signature Parameters Validation:



The photograph should be in the size of 4 cm x 2 cm or (40 mm x 20 mm) and the resolution of the file should be maximum 900 pixels (Width) X 300 pixels (Height). Bit Depth of image file should be 24 bit and DPI range should be between 200 and 300.

Project Set-up

System Requirements:

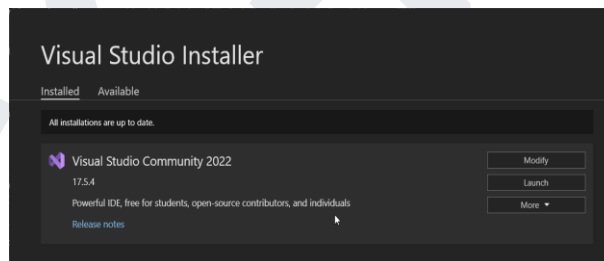
- Windows 64 bit
- Python > 3
- Cmake
- Visual Studio

Downloads:

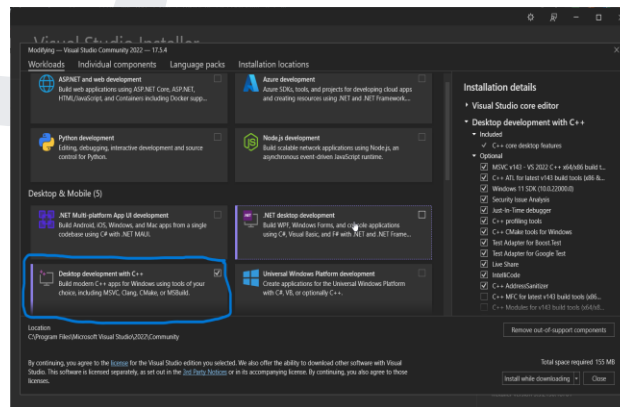
Python	https://www.python.org/downloads/
CMake	cmake-3.26.3-windows-x86_64.msi
Visual Studio	https://visualstudio.microsoft.com/downloads/



- Open visual studio installer and select modify option as shown in below image.



- Select option shown in below screenshot, and install



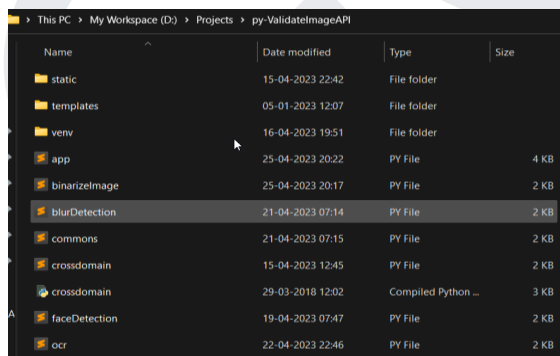
Once above setup is complete install below mentioned python packages using **pip** command line.

Example command: CMD> pip install <package-name>

Python packages / libraries:

- flask
- waitress
- requests
- face_recognition
- cmake
- opencv-python
- pillow
- numpy
- imutils
- scikit-image
- flask-swagger-ui

Project Structure:

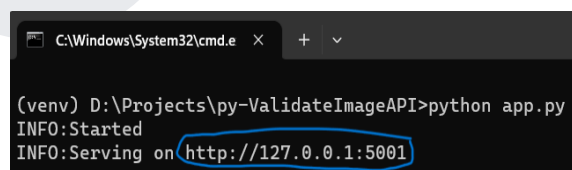


Name	Date modified	Type	Size
static	15-04-2023 22:42	File folder	
templates	05-01-2023 12:07	File folder	
venv	16-04-2023 19:51	File folder	
app	25-04-2023 20:22	PY File	4 KB
binarizeImage	25-04-2023 20:17	PY File	2 KB
blurDetection	21-04-2023 07:14	PY File	2 KB
commons	21-04-2023 07:15	PY File	2 KB
crossdomain	15-04-2023 12:45	PY File	2 KB
crossdomain	29-03-2018 12:02	Compiled Python ...	3 KB
faceDetection	19-04-2023 07:47	PY File	2 KB
ocr	22-04-2023 22:46	PY File	2 KB

Open command prompt in the project folder and run the below command to start the application.

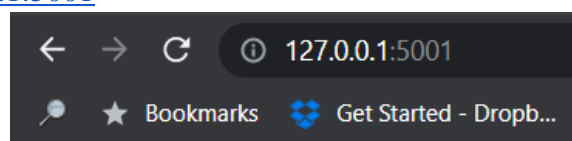
CMD> **python app.py**

If app start is successful, status becomes **Started**.



```
C:\Windows\System32\cmd.e x + v
(venv) D:\Projects\py-ValidateImageAPI>python app.py
INFO:Started
INFO:erving on http://127.0.0.1:5001
```

Navigate to the url <http://127.0.0.1:5001>



Welcome to image validation Application!

API Details

Api	http://127.0.0.1:5001/validateImage
Method	POST
Request Type	application/json
Response Type	application/json
Swagger	http://127.0.0.1:5001/swagger/
Log file location	/<project directory>/static/logs/middleware.log

Request Params:

Param	Type	Value
imageFilePath	String	“D:/detecting_blur_result_006.jpg” or “http://url.com/006.jpg”
id	String	"TS-2023454"
fileType	String	face/sign
appid	String	"TSPSC"

Response Params:

Param	Type	Value
appid	String	TSPSC
createdatetime	String	2023-04-25 20:24
id	String	TS-2023454
isblur	String	True / false
isvalidimage	String	True / false
status	String	OK / ERROR
type	String	Face / sign
uuid	String	0370d2b7-e379-11ed-a2f9-005056c00008

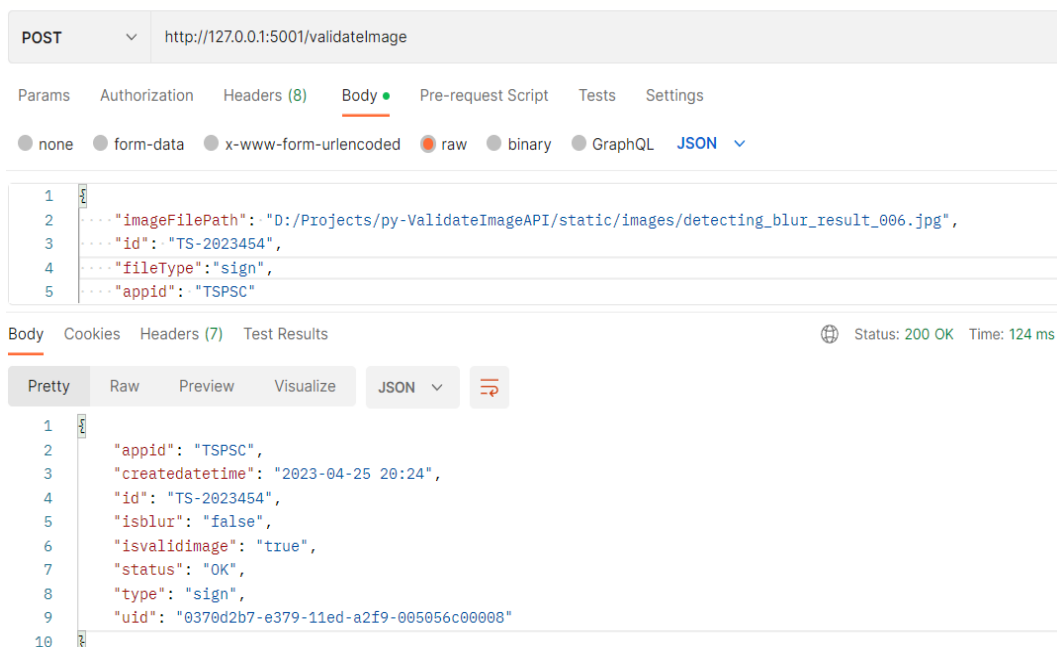
Request Body:

```
{
  "imageFilePath": "D:/Projects/py-ValidateImageAPI/static/images/006.jpg",
  "id": "TS-2023454",
  "fileType": "sign",
  "appid": "TSPSC"
}
```

Response:

```
{
  "appid": "TSPSC",
  "createdatetime": "2023-04-25 20:24",
  "id": "TS-2023454",
  "isblur": "false",
  "isvalidimage": "true",
  "status": "OK",
  "type": "sign",
  "uid": "0370d2b7-e379-11ed-a2f9-005056c00008"
}
```

Sample Postman Request Screenshot



Api Integration Examples

JAVA

1. okHttp library

```
OkHttpClient client = new OkHttpClient().newBuilder().build();

MediaType mediaType = MediaType.parse("application/json");

RequestBody body = RequestBody.create(mediaType, "{\"imageFilePath\": \"D:/Projects/py-ValidateImageAPI/static/images/detecting_blur_result_006.jpg\", \"id\": \"TS-2023454\", \"fileType\": \"sign\", \"appid\": \"TSPSC\"}");

Request request = new Request.Builder() .url("http://127.0.0.1:5001/validateImage") .method("POST", body) .addHeader("Content-Type", "application/json") .build();

Response response = client.newCall(request).execute();
```

2. Unirest Library

```
Unirest.setTimeouts(0, 0);

HttpResponse<String> response = Unirest.post("http://127.0.0.1:5001/validateImage")

    .header("Content-Type", "application/json").body("{ \"imageFilePath\": \"D:/Projects/py-ValidateImageAPI/static/images/detecting_blur_result_006.jpg\", \"id\": \"TS-2023454\", \"fileType\": \"sign\", \"appid\": \"TSPSC\" }").asString();
```

Java Script

```
var myHeaders = new Headers();

myHeaders.append("Content-Type", "application/json");

var raw = JSON.stringify({

  "imageFilePath": "D:/Projects/py-ValidateImageAPI/static/images/detecting_blur_result_006.jpg",

  "id": "TS-2023454",

  "fileType": "sign",

  "appid": "TSPSC"

});

var requestOptions = {

  method: 'POST',

  headers: myHeaders,

  body: raw,

  redirect: 'follow'

};

fetch("http://127.0.0.1:5001/validateImage", requestOptions)

  .then(response => response.text())

  .then(result => console.log(result))

  .catch(error => console.log('error', error));
```

JQuery

```
var settings = { "url": "http://127.0.0.1:5001/validateImage",  
  "method": "POST", "headers": {  
    "Content-Type": "application/json"  
  }, "data": JSON.stringify({  
    "imageFilePath": "D:/Projects/py-ValidateImageAPI/static/images/detecting_blur_result_006.jpg",  
    "id": "TS-2023454", "fileType": "sign",  
    "appid": "TSPSC"  
  }),  
};  
$.ajax(settings).done(function (response) {  
  console.log(response);  
});
```

Python

```
import requests

import json

url = "http://127.0.0.1:5001/validateImage"

payload = json.dumps({ "imageFilePath": "D:/Projects/py-
ValidateImageAPI/static/images/detecting_blur_result_006.jpg",
    "id": "TS-2023454",
    "fileType": "sign",
    "appid": "TSPSC"
})

headers = {
    'Content-Type': 'application/json'
}

response = requests.request("POST", url, headers=headers, data=payload)

print(response.text)
```

