

## Documentation For Face API

### API Name: Microsoft Azure Face API and Computer Vision API

This project uses Azure Face API for Face detection and Computer Vision API for Image analysis. The Azure Face service provides AI algorithms that detect, recognize, and analyze human faces in images. Facial recognition software is important in many different scenarios, such as identity verification, touchless access control, and face blurring for privacy.

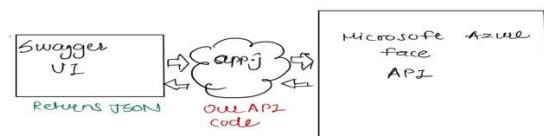
Please refer: <https://learn.microsoft.com/en-us/azure/cognitive-services/computer-vision/overview-identity>

Azure's Computer Vision service gives you access to advanced algorithms that process images and return information based on the visual features. The Image Analysis service extracts many visual features from images, such as objects, faces, adult content, and auto-generated text descriptions.

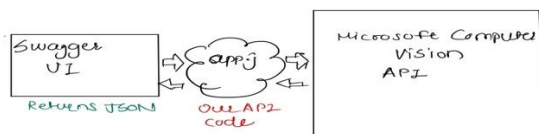
Please refer: <https://learn.microsoft.com/en-us/azure/cognitive-services/computer-vision/overview>

### How it works?

Azure Face API



Computer Vision API



### Required Technologies:

- Node.js
- Postman

### Setup

- Create a free Azure account.
- Create a FaceAPI resource and Computer vision resource and get subscription key and endpoint.

## Installation

- git clone
- cd Final-Project

User needs to create a .env file and paste following data

subscriptionKeyForFaceAPI= <your api key>

endpointForFaceAPI="https://facedemoshruti04.cognitiveservices.azure.com/face/v1.0/detect"

subscriptionKeyForCVAPI= <your api key>

endpointForCV="https://cvdemoshruti04.cognitiveservices.azure.com/vision/v3.2/analyze"

## Install dependencies

- npm install

## Start the application

- node app.js

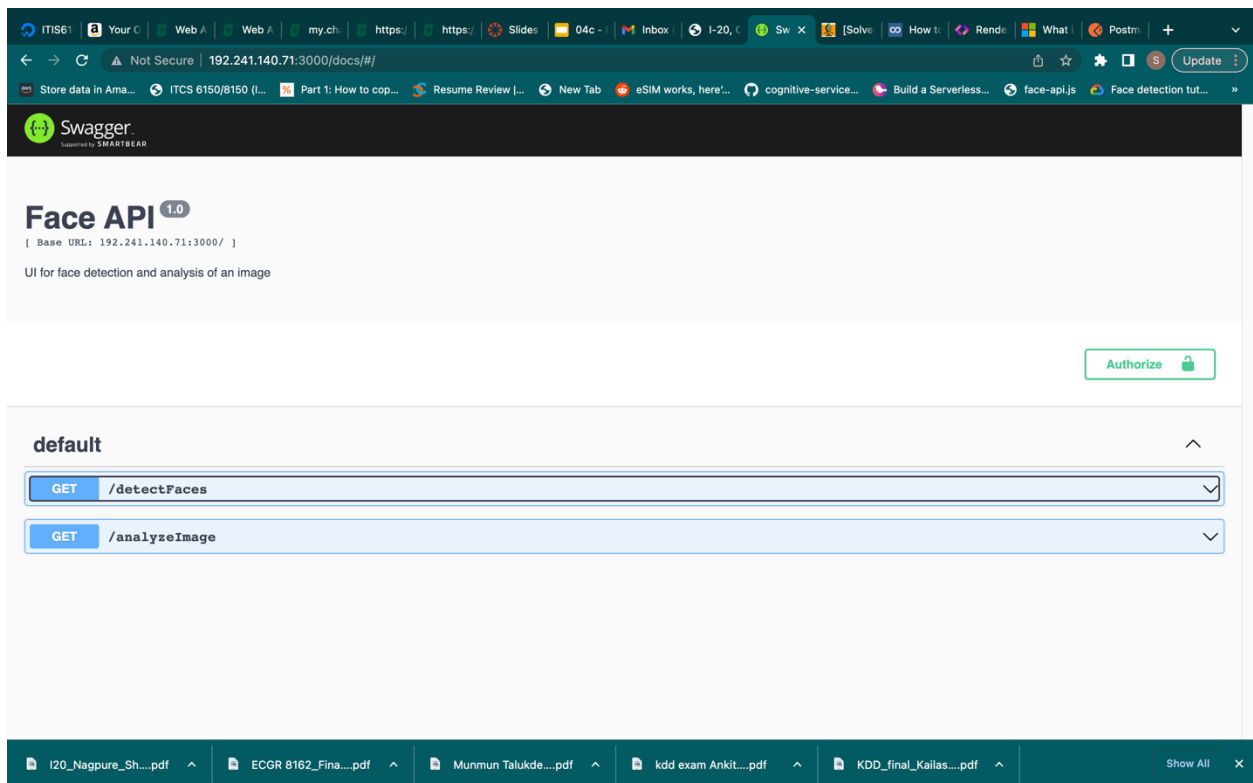
## Testing and Using application

It can be tested on Postman at <http://localhost:3000/>

Usage: - There are two endpoints for this 2 API.

1. /detectFaces
2. /analyzeImage

To test application using swagger UI please access <http://localhost:3000/docs>



## /detectFaces

This is an GET API which gives the faces detected in the given image in the form of rectangle values which is top, left, width and height. User needs to pass the imageUrl to get the response.

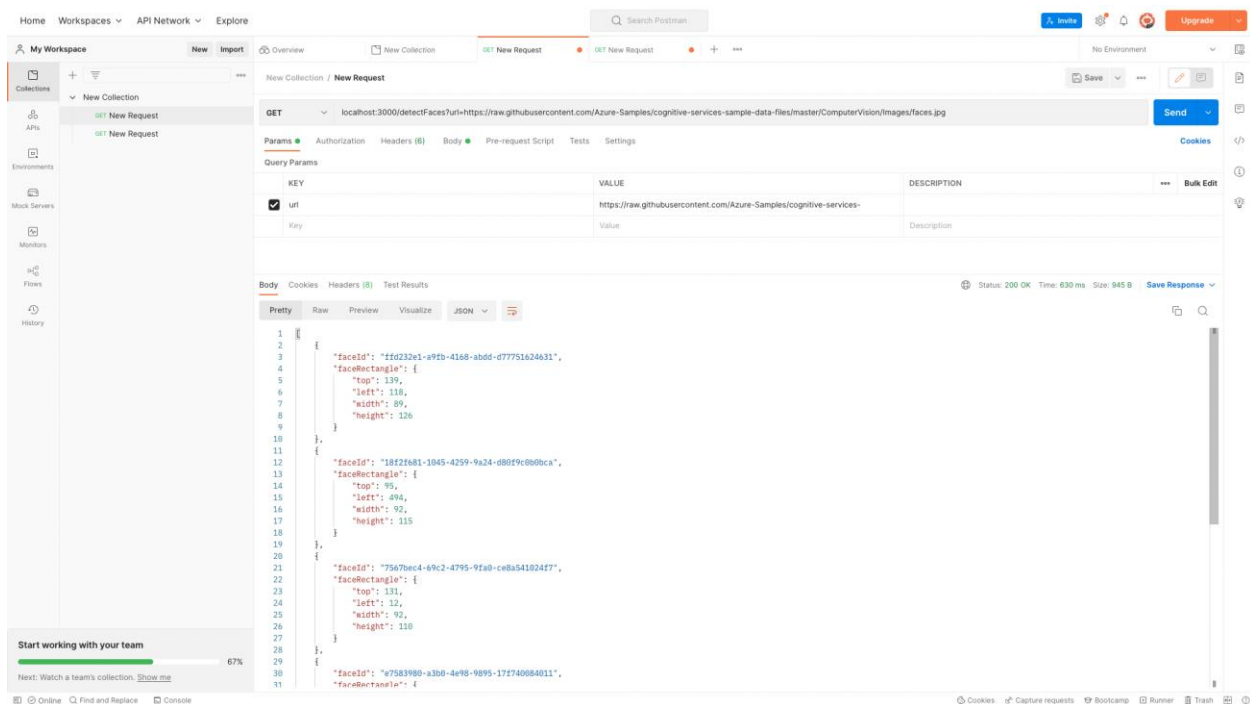
### Image requirement:

Image should be in the form of required format which is jpg,bmp,jpeg.

Image should be less than 4MB.

Dimensions of the image must be greater than 50 x 50 pixels and less than 16,000 x 16,000 pixels.

When user pass the required imageUrl in the request, server code calls the Azure face API and we can see the pixel values of the faces in the given image.

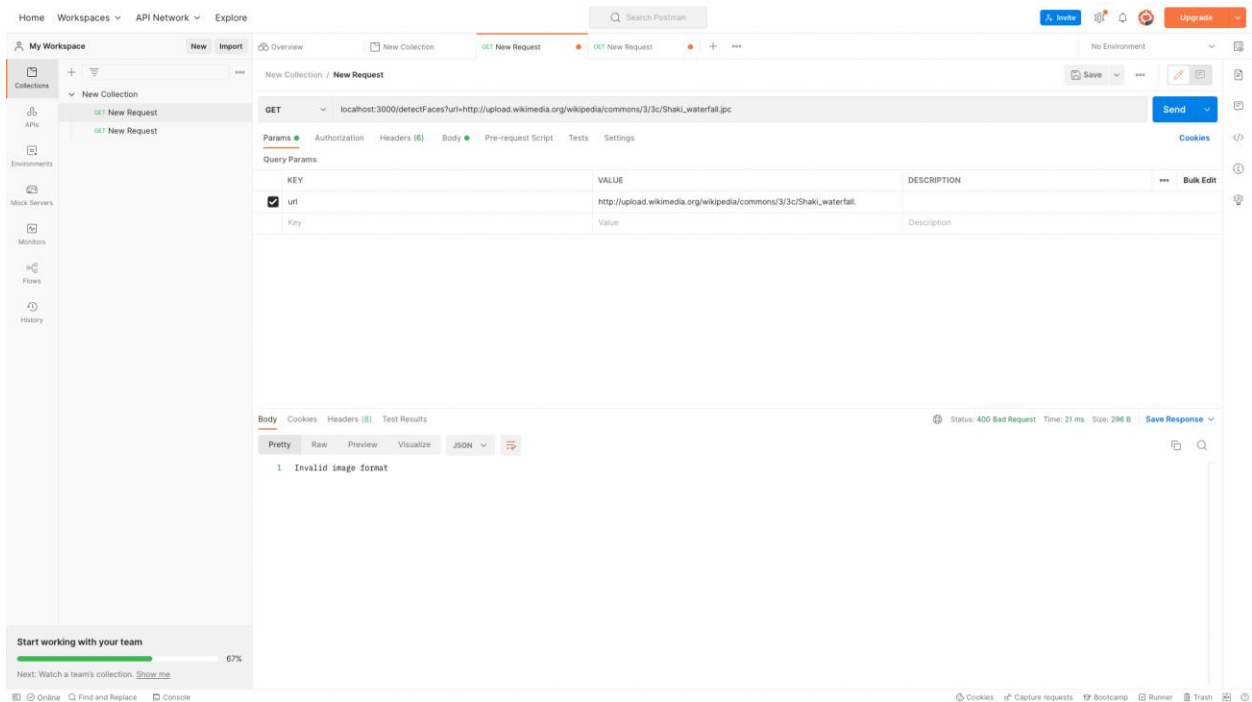
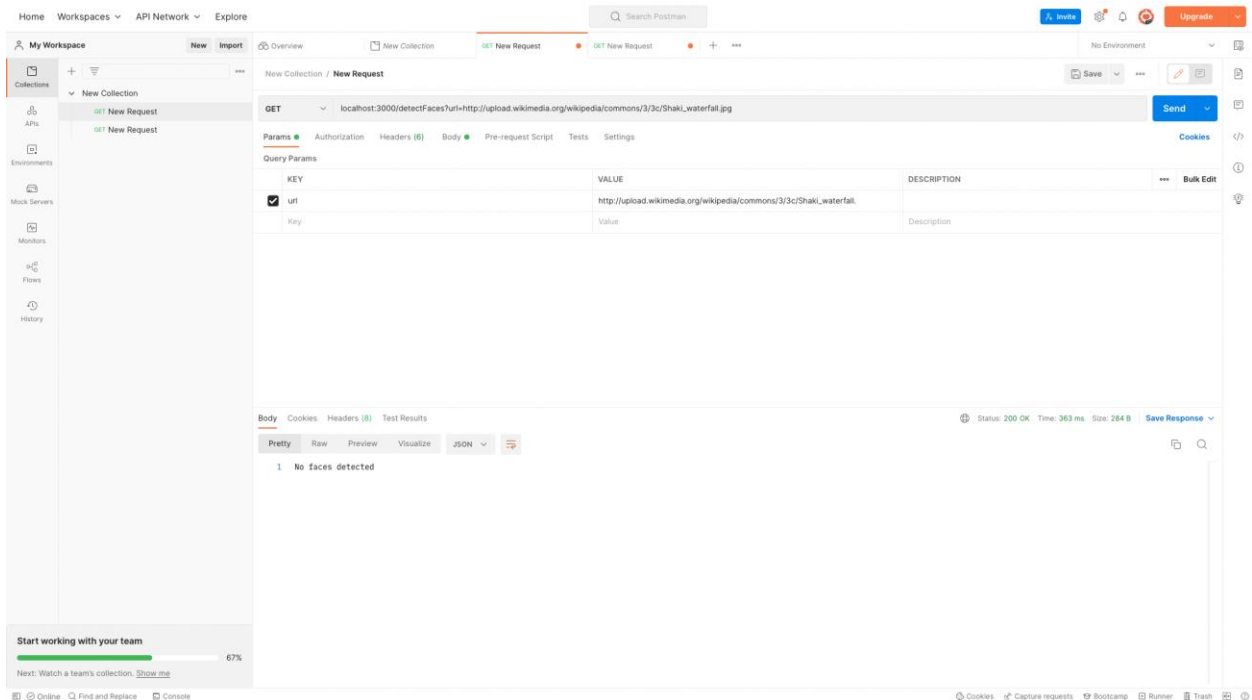


The screenshot shows a Postman interface with a GET request to `localhost:3000/detectFaces?url=https://raw.githubusercontent.com/Azure-Samples/cognitive-services-sample-data-files/master/ComputerVision/Images/faces.jpg`. The request has a query parameter `url` with the value `https://raw.githubusercontent.com/Azure-Samples/cognitive-services-sample-data-files/master/ComputerVision/Images/faces.jpg`. The response is a JSON array of three face objects, each containing `faceId` and `faceRectangle` (top, left, width, height).

```
1 {
2   {
3     "faceId": "1f0232e1-a9fb-4168-ab6d-d77751624631",
4     "faceRectangle": {
5       "top": 139,
6       "left": 116,
7       "width": 89,
8       "height": 126
9     }
10  },
11  {
12    "faceId": "102f28681-1045-4259-9a24-080f9c880bca",
13    "faceRectangle": {
14      "top": 95,
15      "left": 494,
16      "width": 92,
17      "height": 115
18    }
19  },
20  {
21    "faceId": "7567bec4-69c2-4795-9fa0-c6ba541024f7",
22    "faceRectangle": {
23      "top": 131,
24      "left": 12,
25      "width": 92,
26      "height": 110
27    }
28  },
29  {
30    "faceId": "a7563980-a300-4e98-9895-17f740684011",
31    "faceRectangle": {
```

## Handling edge cases:

If user pass the invalid image format or faces with no image then server code validates the image and shows below error.



## /analyzeImage

This is an GET API which gives the analysis of the given image in the form of description. User needs to pass the imageUrl to get the response. The server code uses Azure's computer vision API.

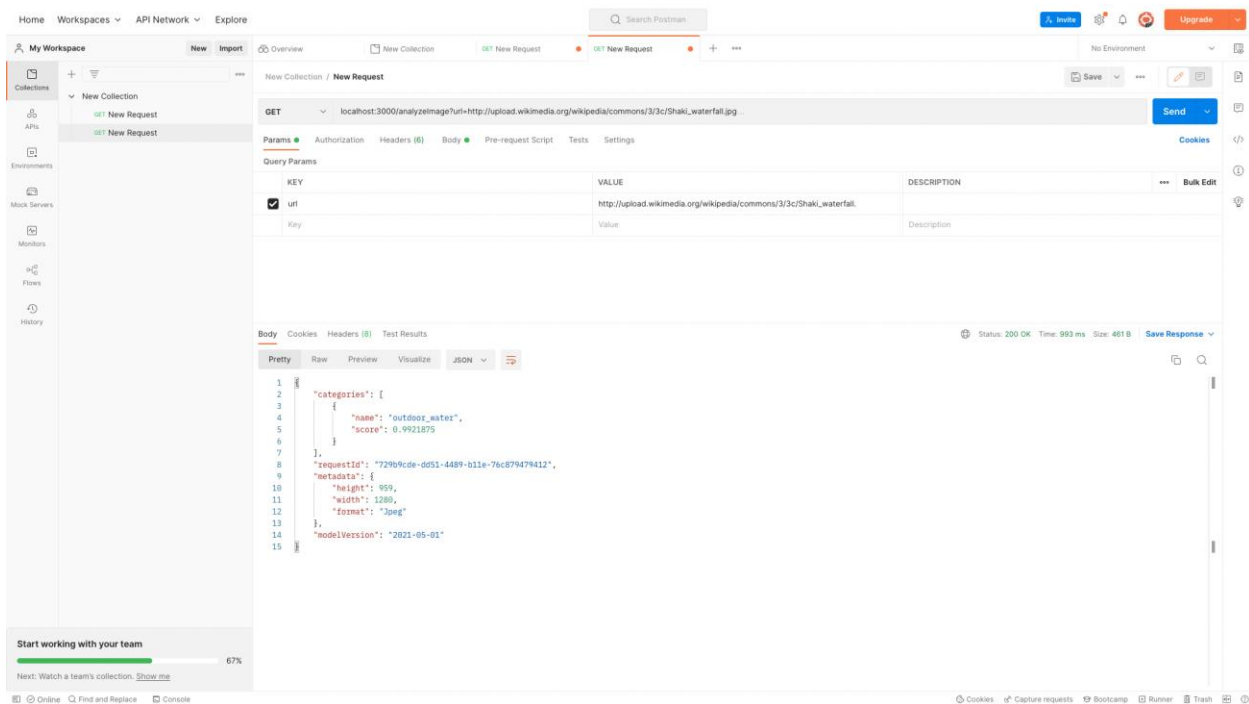
### Image requirement:

Image should be in the form of required format which is jpg,bmp,jpeg.

Image should be less than 4MB.

Dimensions of the image must be greater than 50 x 50 pixels and less than 16,000 x 16,000 pixels.

When user pass the required imageUrl in the request, server code calls the Azure Computer vision API and we can see the description of the given image. It includes name and metadata for the given image in JSON format.



## Passing the faces image:

The screenshot shows the Postman interface with a new request configured. The URL is `localhost:3000/analyzeimage?url=https://raw.githubusercontent.com/Azure-Samples/cognitive-services-sample-data-files/master/ComputerVision/Images/faces.jpg`. The request is a GET method. The response is a JSON object with the following structure:

```
1 {
2   "categories": [
3     {
4       "name": "people_group",
5       "score": 0.9765625
6     }
7   ],
8   "requestId": "f2b054d2-7ef9-439b-a962-2843846d4267",
9   "metadata": {
10     "height": 463,
11     "width": 609,
12     "format": "jpeg"
13   },
14   "modelVersion": "2021-05-01"
15 }
```

The status is 200 OK, Time: 390 ms, Size: 459 B.

## Handling edge cases:

If user pass the invalid image format or faces with no image then server code validates the image and shows below error.

The screenshot shows the Postman interface with a new request configured. The URL is `localhost:3000/analyzeimage?url=http://upload.wikimedia.org/wikipedia/commons/3/3c/Shaki_waterfall.jpg`. The request is a GET method. The response is a 400 Bad Request error with the message "Invalid image format".

```
1 Invalid image format
```

The status is 400 Bad Request, Time: 17 ms, Size: 296 B.

## Reference:

[Microsoft](#)

[Azure API](#)

[Swagger](#)

[Node.js](#)

[Postman](#)