API DOCUMENTATION

Overview:

This documentation illustrates a Python web application named “Render Detection API” that is running on Render (Cloud Computing Platform). The application or API is intended to analyze facial images using machine-learning technology and generate predictions: the probability of depression.

API Endpoints:

The endpoint uses the Fast API library and can be accessed using HTTP GET or POST URL requests. The GET request simply returns a dictionary with a greeting message: "Hi there": "You reached the endpoint!". The POST request expects a base64-encoded image in the body of the request and passes it to the predit_depression() function.

The predit_depression() function takes the base64-encoded image as input, preprocesses it, and uses a TensorFlow model to predict the probability of depression. If the probability is less than 0.01, the function returns a “<0.01” prediction. Otherwise, it rounds the probability to two decimal places and returns it as a string.

The preprocess_image() function takes the base64-encoded image as input, decodes it, converts it to a grayscale image, detects faces in the image using the Haar Cascade classifier, crops and resizes the image to 100x100 pixels, normalizes it, and expands the dimensions of the image to match the input dimensions of the TensorFlow model.

Parameters:

The POST request should include a JSON object with the following properties:

Key: String - “image”, Value: String - base64-encoded image


```
{
    "image":"/9j/4AAQSkZJRgABAQAAAQABAAD/
        2wBDAAIBAQEBAQIBAQECAgICAgQDAgICAgUEBAMEBgUGBgYFBgYGBwkIBgcJBwYGCAsICQoKCgoKBggLDAsKDAkKCgr/
        2wBDAQICAgICAgUDAwUKBwYHCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgoKCgr/
        wAARCARWBFYDASIAAhEBAxEB/8QAHwAAAQUBAQEBAQEAAAAAAAAAAAECAwQFBgcICQoL/
        8QAtRAAAgEDAwIEAwUFBAQAAAF9AQIDAAQRBRIhMUEGE1FhByJxFDKBkaEII0KxwRVS0fAkM2JyggkKFhcYGRolJicoKS
        o0NTY3ODk6Q0RFRkdISUpTVFVWV1hZWmNkZWZnaGlqc3R1dnd4eXqDhIWGh4iJipKTlJWWl5iZmqKjpKWmp6ipqrKztLW
        2t7i5usLDxMXGx8jJytLT1NXW19jZ2uHi4+Tl5ufo6erx8vP09fb3+Pn6/
        8QAHwEAAwEBAQEBAQEBAQAAAAAAAAECAwQFBgcICQoL/
        8QAtREAAgECBAQDBAcFBAQAAQJ3AAECAxEEBSExBhJBUQdhcRMiMoEIFEKRobHBCSMzUvAVYnLRChYkNOEl8RcYGRomJy
        gpKjU2Nzg5OkNERUZHSElKU1RVVldYWVpjZGVmZ2hpanN0dXZ3eHl6goOEhYaHiImKkpOUlZaXmJmaoqOkpaanqKmqsrO
        0tba3uLm6wsPExcbHyMnK0tPU1dbX2Nna4uPk5ebn6Onq8vP09fb3+Pn6/
        9oADAMBAAIRAxEAPwD9VgobqeabRkjkUAEDPPNeZbU9AfkkZUUozjmkJCihTuHIoasAuQe9FNC7W49KdRa6uBImDjJqSo
        lJxyKeCWP0oSYEpXcAc0m1lPFCvjg07qcVTbSIFQ4PWn0xlI4FOUYFCdxDmUAAqc
```


Responses:

- 200 OK: The API will return this response when the request is successful, and the app is able to analyze the image data provided by the user. The response will include a JSON object with the following properties:

  Key: String - “Prediction”

  Value: String – A score between 0 and 1 indicating the probability of depression

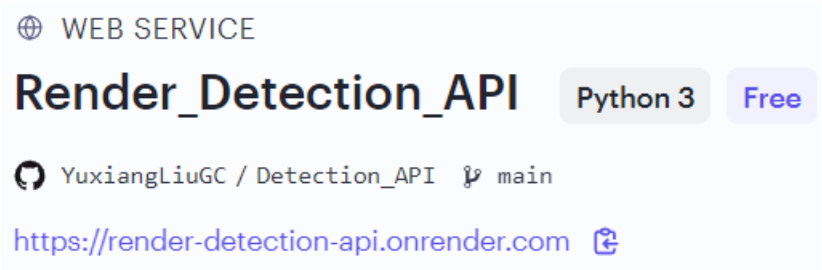
```
{
    "prediction": "0.72"
}
```

- 500 Internal Server Error: The API will return this response when there is an error on the server side, or the image data was corrupted.

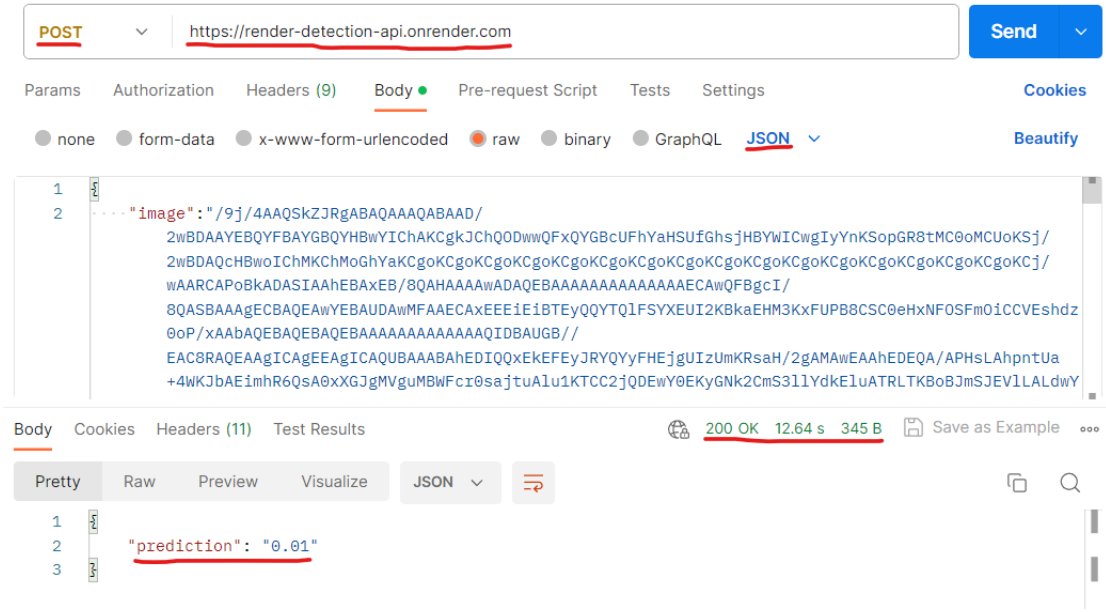
Render

Note that the API is running with a free plan so it will spin down with inactivity. To prevent this behavior upgrading to a paid instance type is needed.

[Link to manage the API]



- Example of making a request:



-