

TOPIC : AI powered nutrition analyzer for fitness enthusiast

Team id : PNT2022TMID15800

Routing To The Html Page

Here, the declared constructor is used to route to the HTML page created earlier.

In the above example, the '/' URL is bound with the home.html function. Hence, when the home page of the webserver is opened in the browser, the HTML page is rendered. Whenever you enter the values from the HTML page the values can be retrieved using the POST Method. Here, "home.html" is rendered when the home button is clicked on the UI

```
15
16 @app.route('/')# route to display the home page
17 def home():
18     return render_template('home.html')#rendering the home page
19
20 @app.route('/image1',methods=['GET','POST'])# routes to the index html
21 def image1():
22     return render_template("image.html")
23
```

When "image is uploaded "on the UI, the launch function is executed

```
23
24 @app.route('/predict',methods=['GET', 'POST'])# route to show the predictions in a web UI
25 def launch():
26     if request.method=='POST':
27         f=request.files['file'] #requesting the file
28         basepath=os.path.dirname("__file__")#storing the file directory
29         filepath=os.path.join(basepath,"uploads",f.filename)#storing the file in uploads folder
```

It will take the image request and we will be storing that image in our local system then we will convert the image into our required size and finally, we will be predicting the results with the help of our model which we trained and depending upon the class identified we will showcase the class name and its properties by rendering the respective html pages.

```
20 @app.route('/image1', methods=['GET', 'POST'])# routes to the index html
21 def image1():
22     return render_template("image.html")
23
24 @app.route('/predict', methods=['GET', 'POST'])# route to show the predictions in a web UI
25 def launch():
26     if request.method=="POST":
27         f=request.files["file"] #requesting the file
28         basepath=os.path.dirname(__file__)#storing the file directory
29         filepath=os.path.join(basepath,"uploads",f.filename)#storing the file in uploads folder
30         f.save(filepath)#saving the file
31
32
33         img=image.load_img(filepath,target_size=(64,64)) #load and reshaping the image
34         x=image.img_to_array(img)#converting image to an array
35         x=np.expand_dims(x,axis=0)#changing the dimensions of the image
36
37
38         pred=np.argmax(model.predict(x), axis=1)
39         print("prediction",pred)#printing the prediction
40         index=['APPLES','BANANA','ORANGE','PINEAPPLE','WATERMELON']
41
42
43         result=str(index[pred[0]])
44
45         x=result
46         print(x)
47         result=nutrition(result)
48         print(result)
49
50     return render_template("0.html",showcase=(result),showcase1=(x))
51
52 def nutrition(index):
53     url = "https://calorieninja.p.rapidapi.com/v1/nutrition"
54     querystring = {"query":index}
55
56     headers = {
57         'x-rapidapi-key': "5d737ab18725b668f26bd04465401ff4b34isnf47hf9a8Be4"
```

API Integration:

Here we will be using Rapid API

Using RapidAPI, developers can search and test the APIs, subscribe, and connect to the APIs — all with a single account, single API key and single SDK. Engineering teams also use RapidAPI to share internal APIs and microservice documentation.

[Reference link](#)

API used: [Link](#)

The link above will allow us to test the food item and will result the nutrition content present in the food item.

NOTE: When we keep hitting the API the limit of it might expire. So making a smart use of it will be an efficient way.

How to access and use the API will be shown in this [video](#)

```
File Edit Selection View Go Run Terminal Help
app.py - Visual Studio Code

C:\Users> cd > Downloads > app.py
35 x=np.expand_dims(x,axis=0)#changing the dimensions of the image
36
37
38 pred=np.argmax(model.predict(x), axis=1)
39 print("prediction",pred)#printing the prediction
40 index= ['APPLES', 'BANANA', 'ORANGE', 'PINEAPPLE', 'WATERMELON']
41
42
43 result=str(index[pred[0]])
44
45 x=result
46 print(x)
47 result=nutrition(result)
48 print(result)
49
50 return render_template("0.html",showcase=(result),showcase1=(x))
51
52 def nutrition(index):
53     url = "https://calorieninjas.p.rapidapi.com/v1/nutrition"
54     querystring = {"query":index}
55
56     headers = {
57         'x-rapidapi-key': "5d797ab107mshe668f26bd044e64p1ffd34jsnf47bfa9a8ee4",
58         'x-rapidapi-host': "calorieninjas.p.rapidapi.com"
59     }
60
61     response = requests.request("GET", url, headers=headers, params=querystring)
62
63     print(response.text)
64     return response.json()['items']
65
66 if __name__ == "__main__":
67     # running the app
68     app.run(debug=False)
69
70
71
```

Finally, Run the application

This is used to run the application in a localhost. The local host runs on port number 5000.(We can give different port numbers)

```
64 return response.json()['items']
65
66 if __name__ == "__main__":
67     # running the app
68     app.run(debug=False)
69
70
71
```