

TEAM ID : PNT2022TMID01461

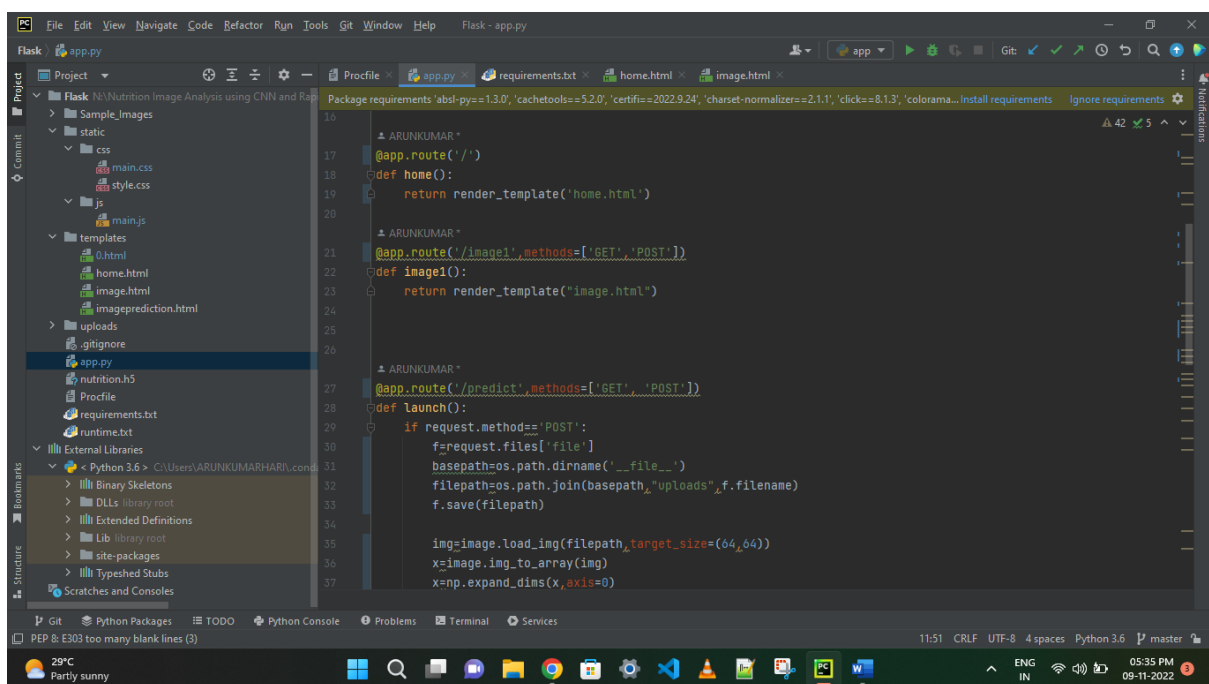
PROJECT NAME : AI-powered Nutrition Analyzer for Fitness Enthusiasts

## 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

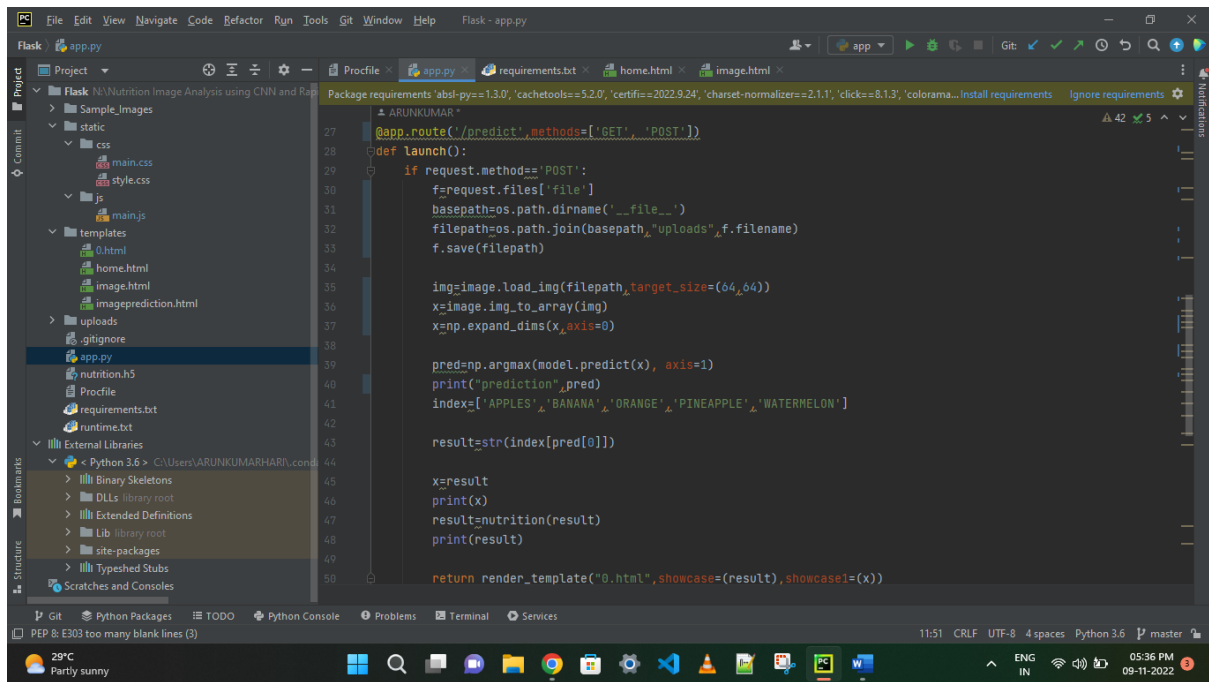


The screenshot shows a code editor with the following code in `app.py`:

```
16  ARUNKUMAR *
17  @app.route('/')
18  def home():
19      return render_template('home.html')
20
21  ARUNKUMAR *
22  @app.route('/image1', methods=['GET', 'POST'])
23  def image1():
24      return render_template("image.html")
25
26  ARUNKUMAR *
27  @app.route('/predict', methods=['GET', 'POST'])
28  def launch():
29      if request.method == 'POST':
30          f = request.files['file']
31          basepath = os.path.dirname('__file__')
32          filepath = os.path.join(basepath, "uploads", f.filename)
33          f.save(filepath)
34
35          img = image.load_img(filepath, target_size=(64, 64))
36          x = image.img_to_array(img)
37          x = np.expand_dims(x, axis=0)
```

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

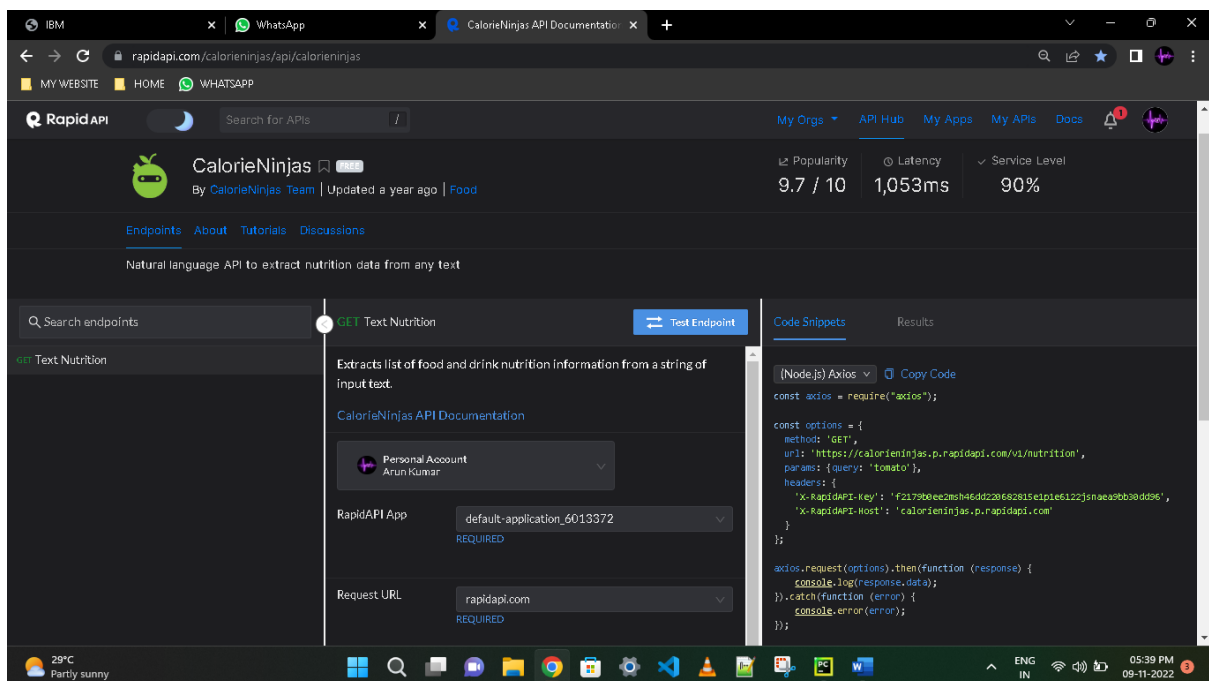
It will take the image request and we will be storing that image in our local system then we will convert the image into our required size and finally, we will be predicting the results with the help of our model which we trained and depending upon the class identified we will showcase the class name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



API Integration:

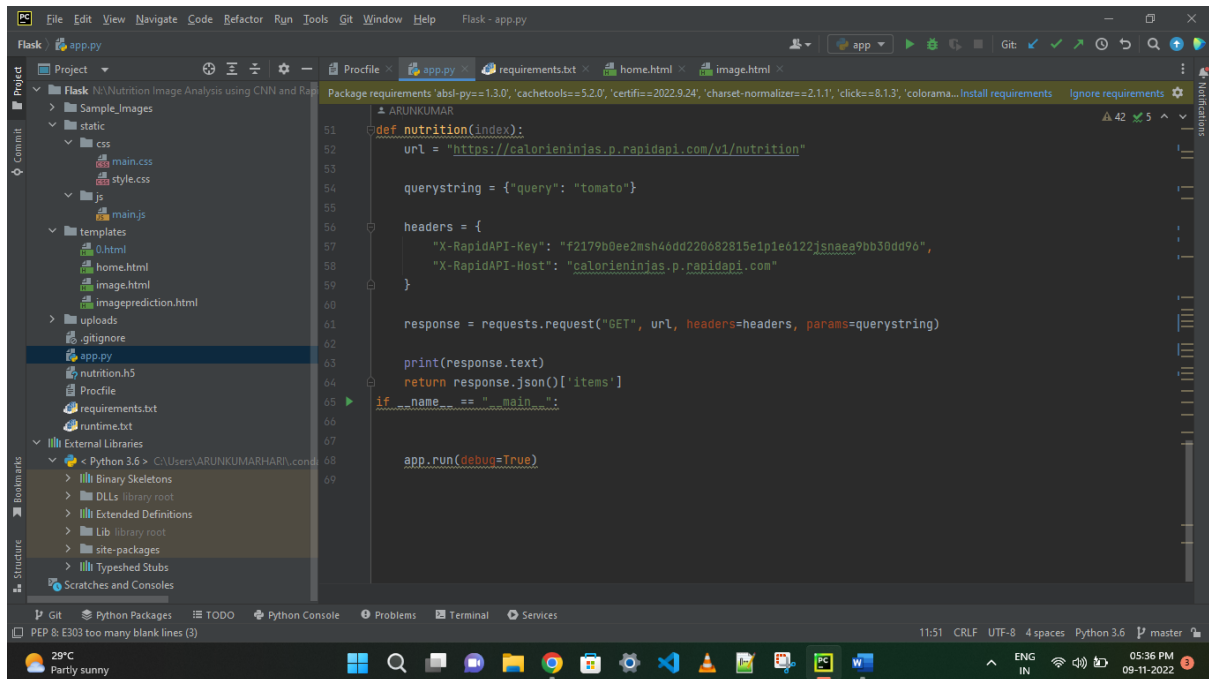
Here we will be using Rapid API

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.



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.



The screenshot shows a Visual Studio Code editor with a project named 'Flask - app.py'. The file explorer on the left shows a directory structure with files like 'main.css', 'style.css', 'main.js', '0.html', 'home.html', 'image.html', 'imageprediction.html', 'uploads', '.gitignore', 'app.py', 'nutrition.h5', 'Profile', 'requirements.txt', and 'runtime.txt'. The main editor window displays the code for 'app.py', which includes a 'def nutrition(index):' function. The function makes a GET request to 'https://calorieninjas.p.rapidapi.com/v1/nutrition' with a querystring of 'tomato' and specific headers. The response is printed and returned as JSON. The code is wrapped in an 'if \_\_name\_\_ == "\_\_main\_\_":' block with 'app.run(debug=True)'. The status bar at the bottom shows '11:51 CRLF UTF-8 4 spaces Python 3.6 master'.

```
51 def nutrition(index):
52     url = "https://calorieninjas.p.rapidapi.com/v1/nutrition"
53
54     querystring = {"query": "tomato"}
55
56     headers = {
57         "X-RapidAPI-Key": "f2179b0ee2msh46dd220682815e1p1e0122jsnaea9bb30dd96",
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
68     app.run(debug=True)
69
```

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)