

**TEAM ID :** PNT2022TMID00751

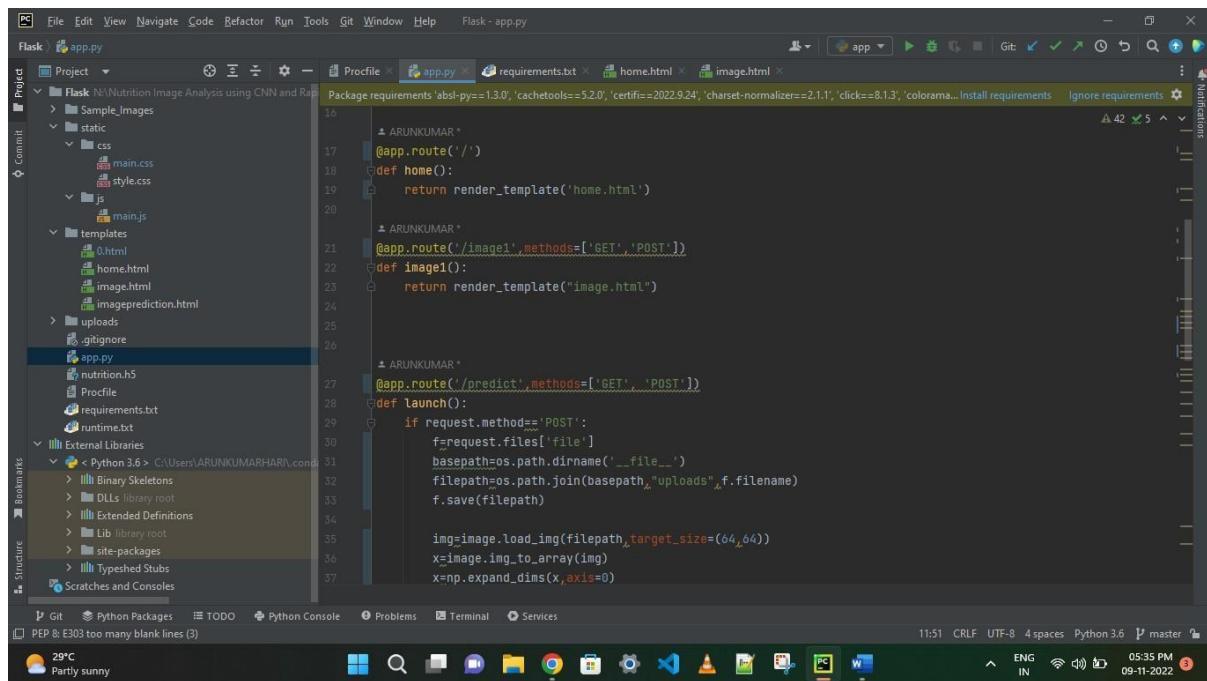
**PROJECT NAME :** AI-powered Nutrition Analyzer for Fitness Enthusiasts

## 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 the PyCharm IDE interface with the following details:

- Project Structure:** The project is named "Flask NINutrition Image Analysis using CNN and RNN". It contains a "static" folder with CSS and JS files, a "templates" folder with HTML files like "home.html" and "image.html", and an "uploads" folder. There are also "nutrition.h5", "Procfile", "requirements.txt", and "runtime.txt" files.
- Code Editor:** The file "app.py" is open, showing Python code for a Flask application. The code defines routes for the home page, an image viewer, and a prediction endpoint. It uses the "render\_template" function to serve HTML pages and the "load\_img" and "img\_to\_array" functions to handle image requests.
- Toolbars and Status Bar:** The status bar at the bottom shows the current environment (Python 3.6), master branch, and system information like date and time.

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.

The screenshot shows the PyCharm IDE interface with the following details:

- Project Structure:** The project is named "Flask NINutrition Image Analysis using CNN and Rap". It contains a "static" folder with CSS files (main.css, style.css), a "templates" folder with HTML files (0.html, home.html, image.html, imageprediction.html), and an "uploads" folder. A ".gitignore" file is also present.
- Code Editor:** The main editor window displays the "app.py" file. The code implements a Flask application that handles file uploads and performs image prediction using a pre-trained model. It prints the prediction and its index, then uses a nutrition API to get detailed nutritional information for the predicted item.
- Toolbars and Status Bar:** The top bar includes standard options like File, Edit, View, Navigate, Code, Refactor, Run, Tools, Git, Window, Help, and Flask - app.py. The status bar at the bottom shows the current time (11:51), date (09-11-2022), and system status (ENG IN).

## API Integration:

Here we will be using Rapid API

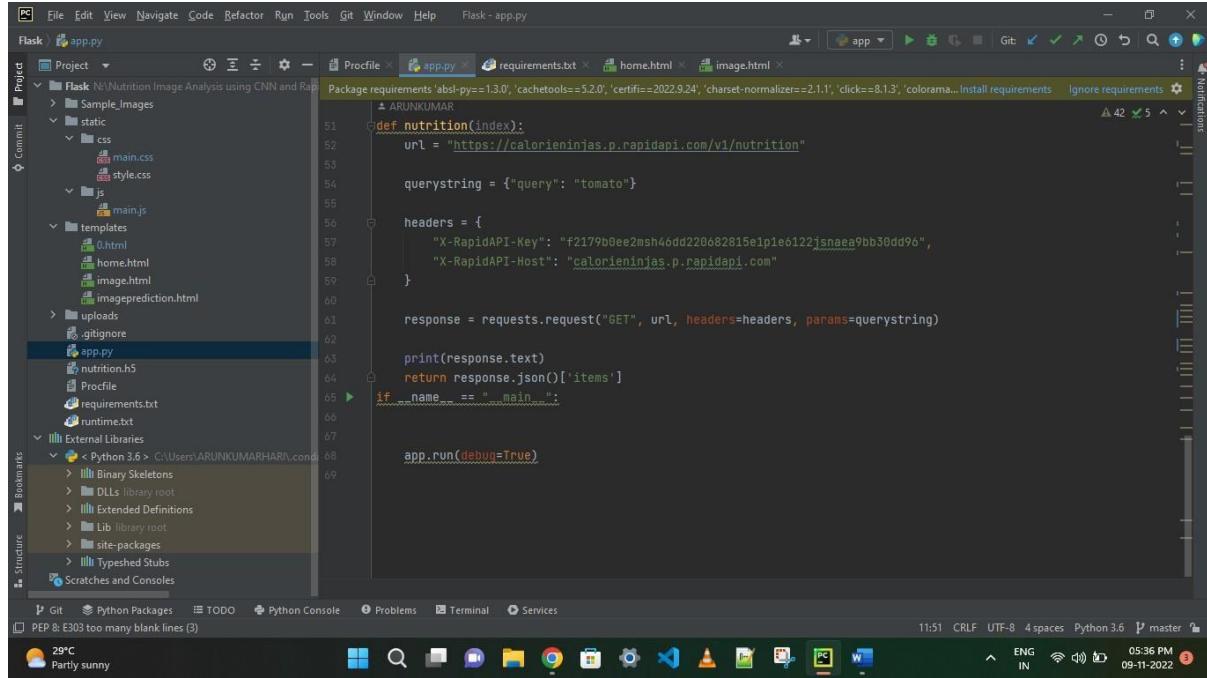
Using RapidAPI, developers can search and test the APIs, subscribe, and connect to the APIs — all with a single account, single API key and single SDK. Engineering teams also use RapidAPI to share internal APIs and microservice documentation.

The screenshot shows the CalorieNinjas API endpoint on the RapidAPI platform. Key details include:

- Popularity:** 9.7 / 10
- Latency:** 1,053ms
- Service Level:** 90%
- Endpoint:** GET Text Nutrition
- Description:** Extracts list of food and drink nutrition information from a string of input text.
- Documentation:** CalorieNinjas API Documentation
- Auth:** Personal Account (Arunkumar)
- Request URL:** rapidapi.com
- Code Snippets:** Node.js Axios code provided for testing.

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 the PyCharm IDE interface with the following details:

- Project Structure:** The project is named "Flask NINutrition Image Analysis using CNN and Rap". It contains a "Sample\_Images" folder, a "static" folder with "css" and "js" subfolders, a "templates" folder with "0.html", "home.html", "image.html", and "imageprediction.html", an "uploads" folder, and a ".gitignore" file. The "app.py" file is selected in the Project view.
- Code Editor:** The "app.py" file is open, showing Python code for a Flask application. The code includes imports for requests, json, and os. It defines a "nutrition" function that sends a GET request to a RapidAPI endpoint for nutrition information on "tomato". It handles the response and returns the items if the name is "main". Finally, it runs the application with debug=True.
- Toolbars and Status Bar:** The status bar at the bottom shows the current time (11:51), encoding (CRLF), file encoding (UTF-8), code style (4 spaces), Python version (Python 3.6), and branch (master). The system tray shows weather (29°C, Partly sunny) and system icons.

## 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)