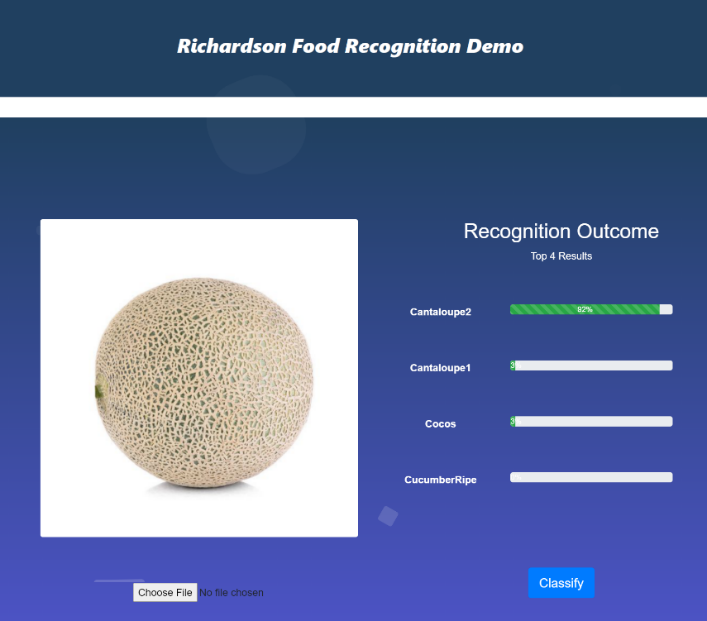
**DEPLOY YOUR CLASSIFIER**

NOTE: We strongly recommend watching our [lesson 5: API Deployment](https://www.youtube.com/watch?v=APUWvsOl86E&feature=youtu.be) before you go through this document. In this vide, Lucas is showing a food classifier that can be accessed by anyone with the URL. In other words, this is a public website.

<http://richardsonfooddemo.azurewebsites.net/>



For simplification purposes, this guide will focus on the basic structure required for you to run the application **locally** (only you can see it). This is a great way for prototyping and debugging before launch.

For those not running the code, this is still a great source to review as it provides an intuitive approach of what is required to have a web app.

Table of Contents

[BASIC SCHEMA 2](#_Toc45275620)

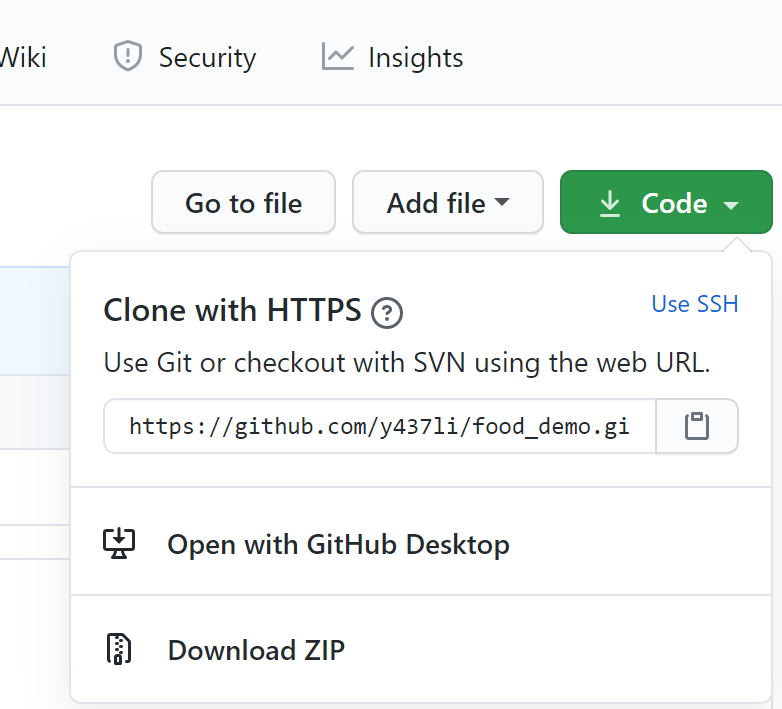
[FRONT END SCRIPT  *image\_rec.html* 3](#_Toc45275621)

[BACK END SCRIPT  *app.py* 5](#_Toc45275622)

[RUN YOUR APPLICATION LOCALLY 7](#_Toc45275623)

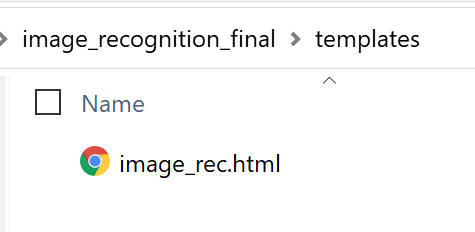
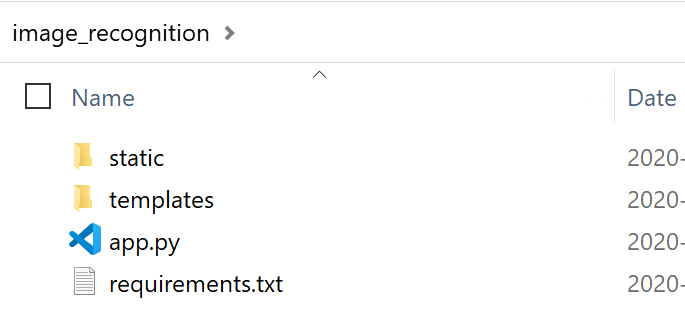
# BASIC SCHEMA

Download the content located in this repo: insert repo url here. Save it in a



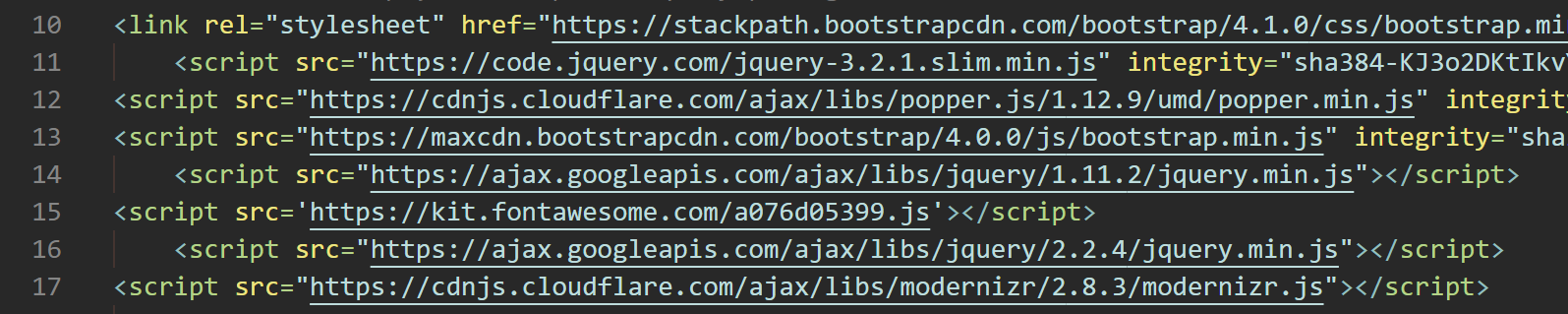
In this case, we have 4 main components:

1. **Front end à** *image\_rec.html* under the templates folder. This file contains text that a web browser requires to render on the screen
2. **Back end à** *app.py*. This file includes any application configuration for the app. In here, we are including info such as API, endpoint, headers, parameters, etc.
3. **Static folderà** This one contains the same pre-built content once we load the page (images, gifs, icons, etc.)
4. **Requirements 🡪** this .txt file provides the list of all the packages/libraries required to run the application. You can install these globally, however, we recommend creating a virtual environment to avoid affecting other projects’ system tools.

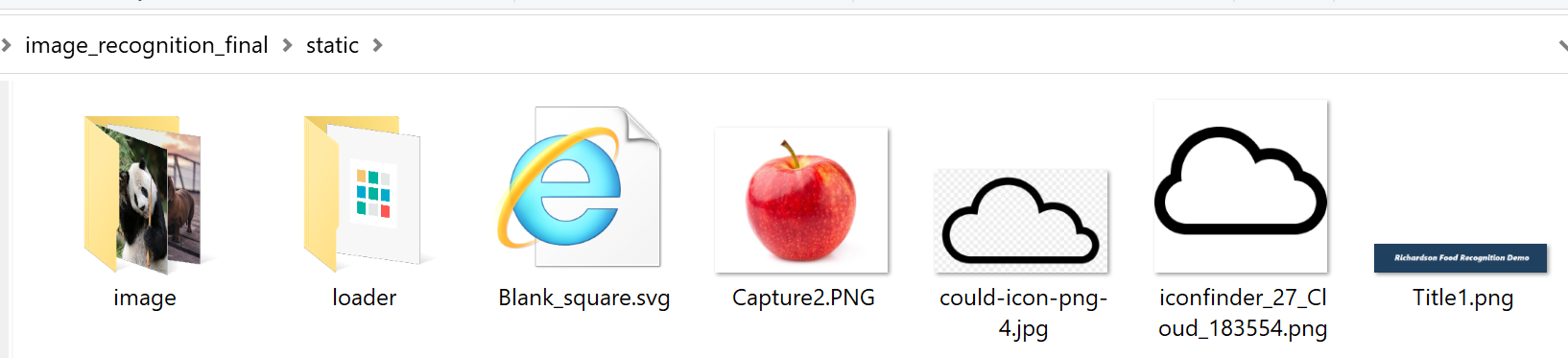


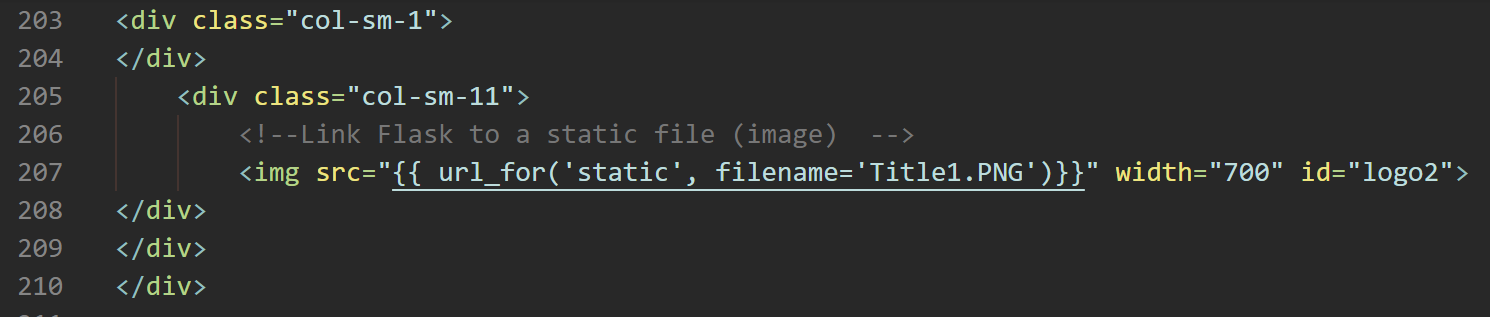
# FRONT END SCRIPT à *image\_rec.html*

**Lines 10 to 17:** JavaScript, bootstrap and jQuery packages are called



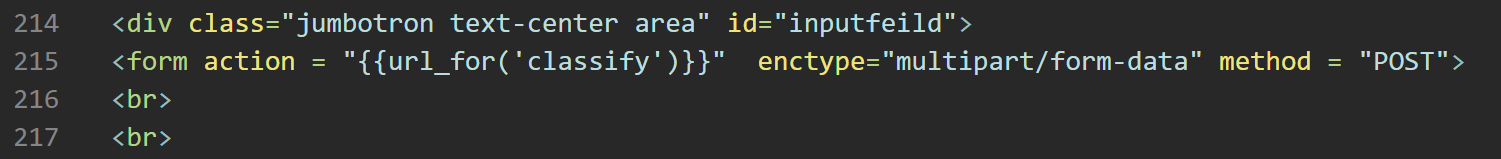
**Lines 17 to 207:** script mostly focused on the design of the page. Here is also where static content is used. For instance, line 207 will load the file *Title1.png* at the top of the website.



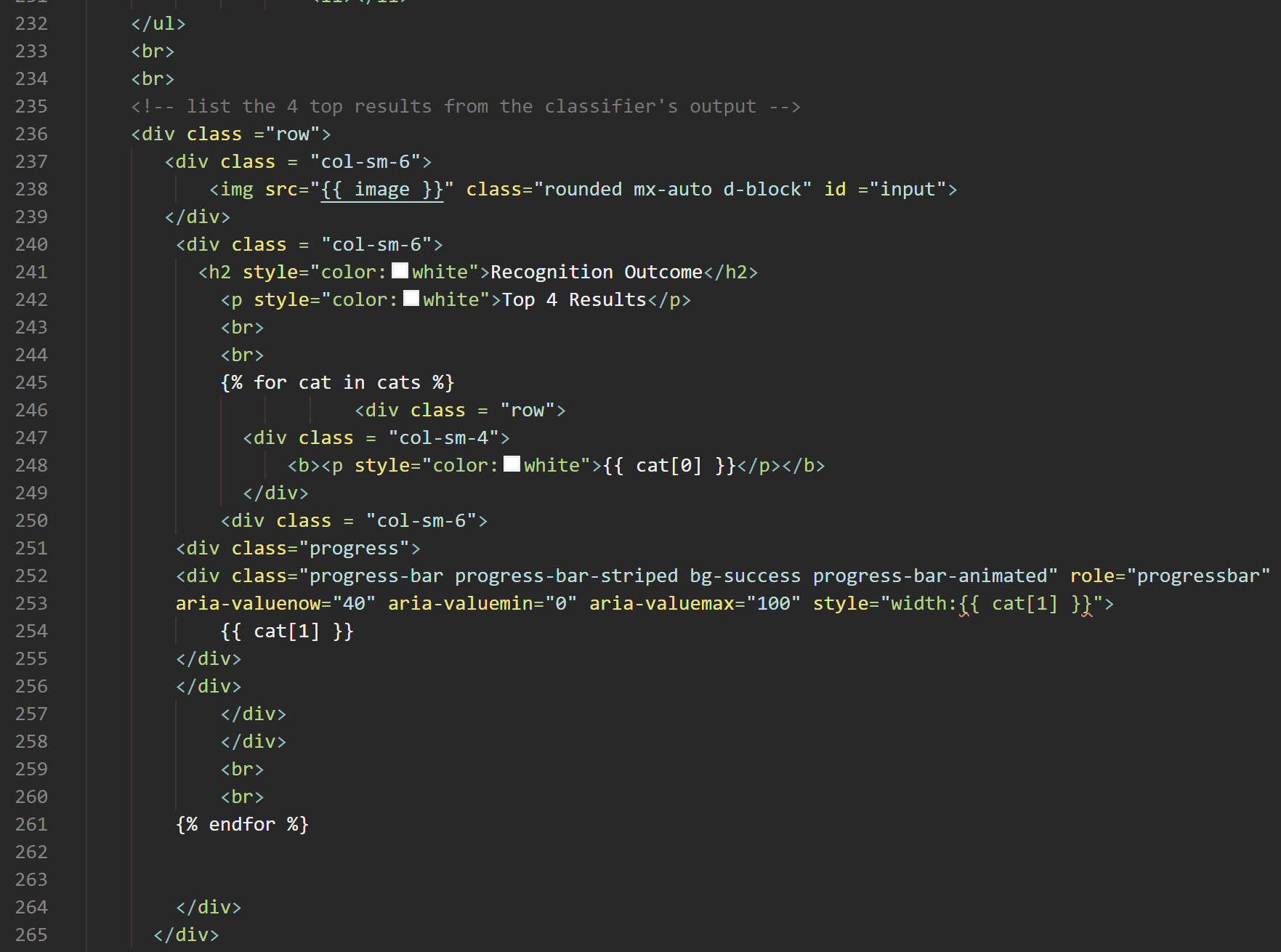
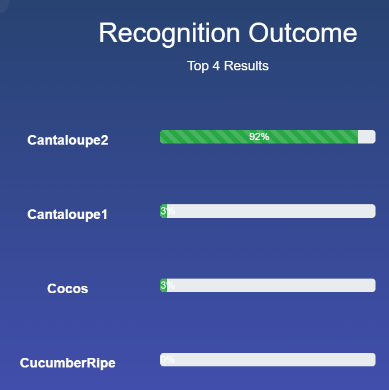




**Lines 214 to 215:** it connects the front end to the 'classify' function defined in the back end. In this case, we are using a POST method: the data (image) is provided from client to server

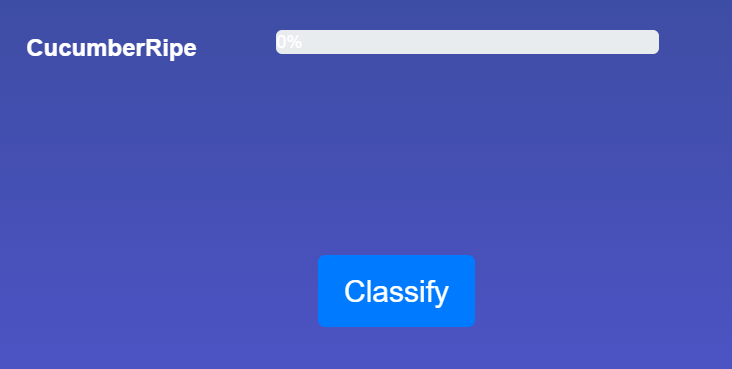


**Lines 236 to 264:** List the top 4 results from the classifier



**Line 278:** Submit the image by clicking the 'Classify' button





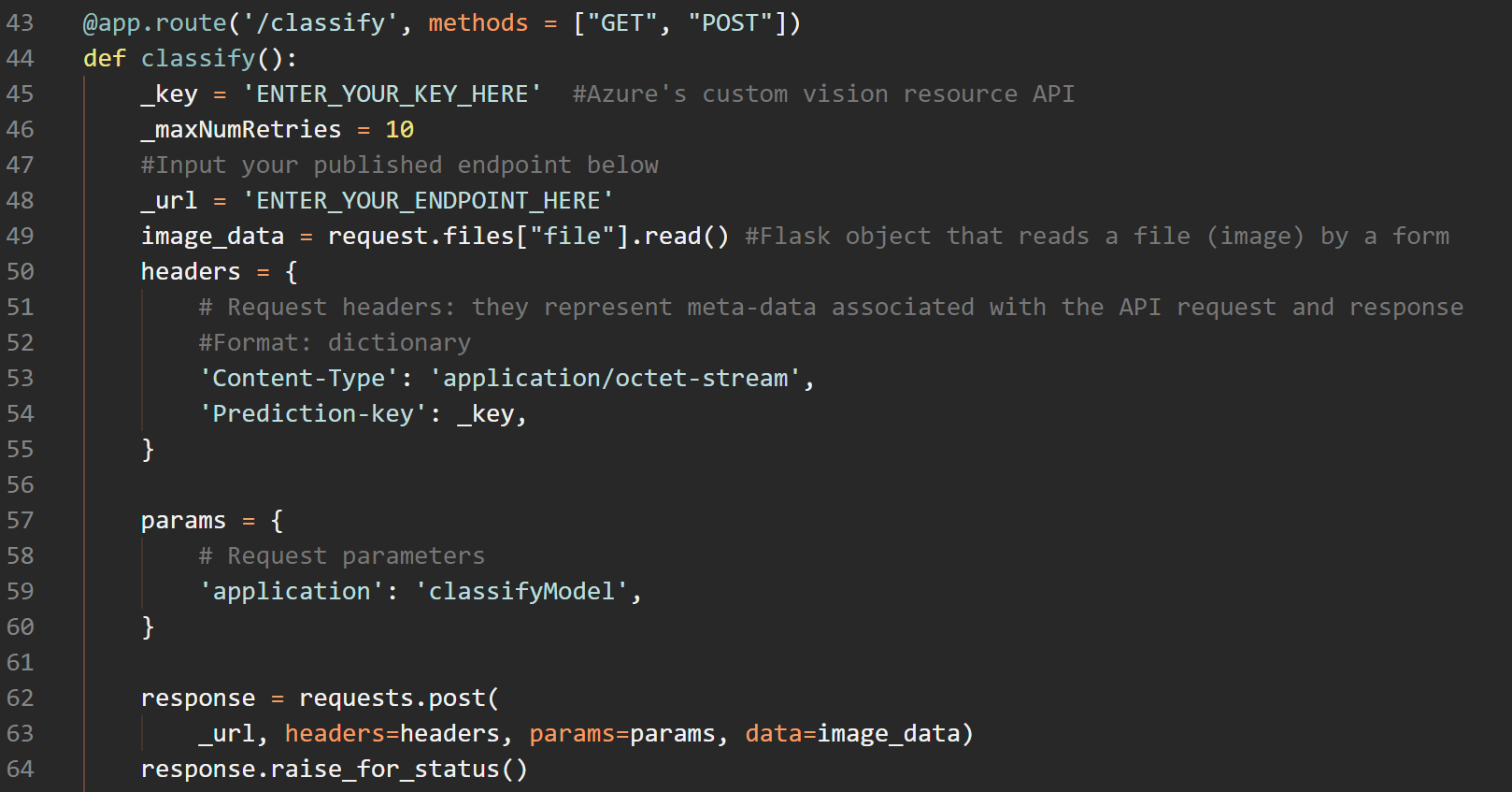
# BACK END SCRIPT à *app.py*

**Lines 43 to 65:** This is the most important part of the Custom Vision API deployment.

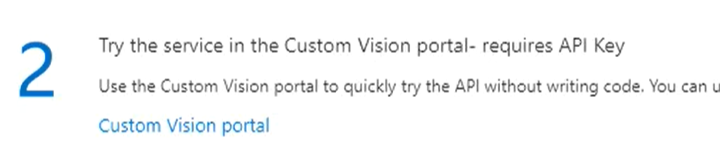
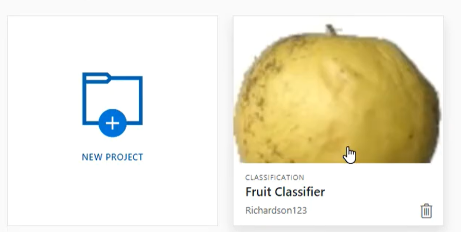
As explained in the video, you are required to have the following items:

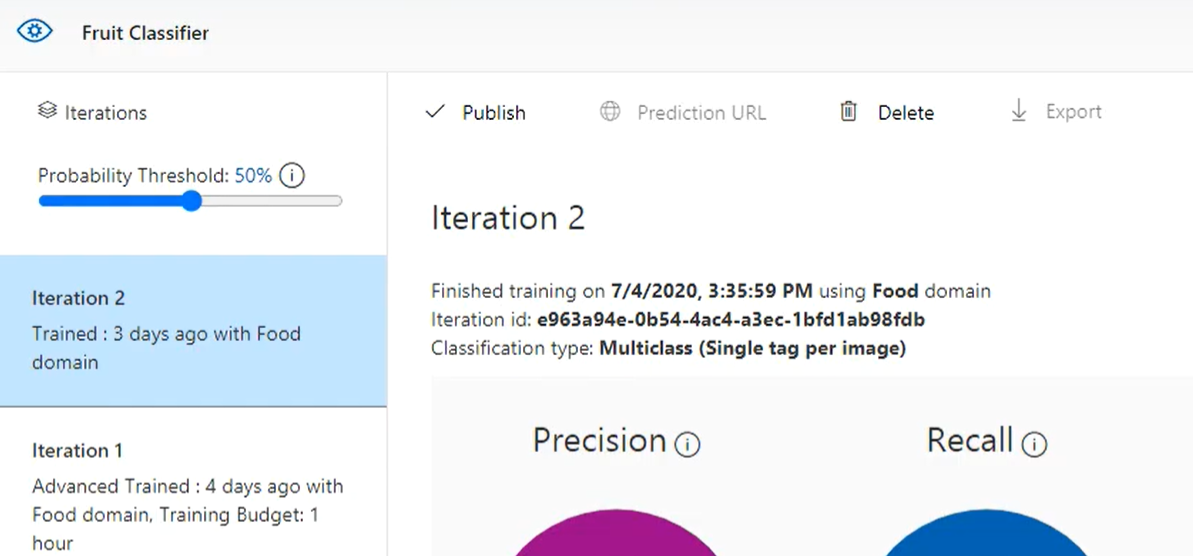
* Key: Unique custom vision resource API
* Endpoint (\_url): [Microsoft’s Custom Vision Services website](https://southcentralus.dev.cognitive.microsoft.com/docs/services/Custom_Vision_Prediction_3.0/operations/5c82db60bf6a2b11a8247c14) provides an in-depth explanation of what is required for your endpoint to work
* Data à The uploaded image
* Headers à meta-data associated with the API request and the response. This is in a dictionary format
* Params à the service we are requesting (“*classifyModel*”)

These items are then sent via POST method to the web server. Once the request has been validated, the Azure’s custom classifier is activated for the input data (image) and produces the output. Keep in mind that you are required to input your key and endpoint in this section (circled in red).

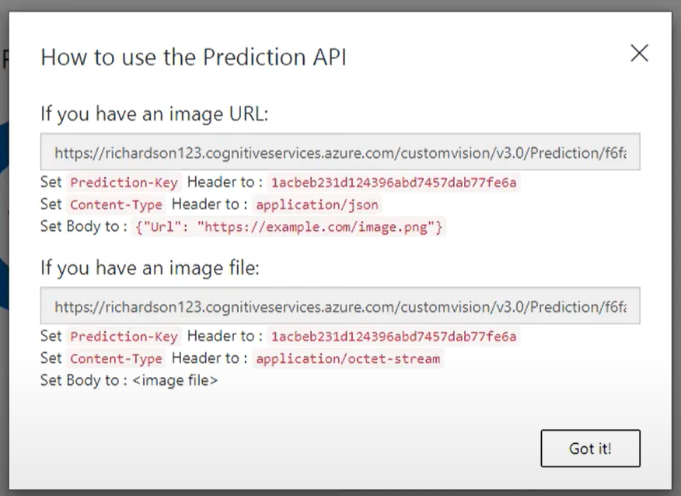


You can find all the required info by following below steps:

1. Go to your custom classifier prediction resource (under the custom vision portal) 
2. Under the performance tab (top side of the portal), select the iteration you want to use and publish it. Select the resource you used for this classifier.



1. Finally, the *Prediction URL* tab will be available. This will provide you with items such as the key, the header, etc.

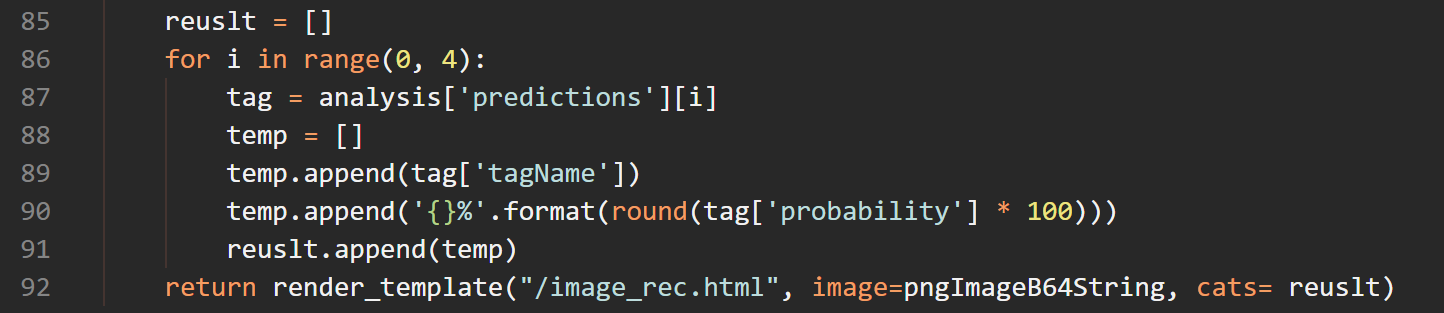


This is your endpoint (**\_url**) under line 48

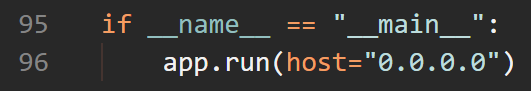
This is your API key (**\_api**) under line 45

This is your **header** under line 53

**Lines 86 to 92:** this section will extract the 4 top results of the classifier which are then pulled and displayed in the front end (*image\_rec.html* file, lines 236 to 264)



**Lines 95 and 96:** make the server (IP) publicly available by using 0.0.0.0 as the host (deployment purposes)



# RUN YOUR APPLICATION LOCALLY

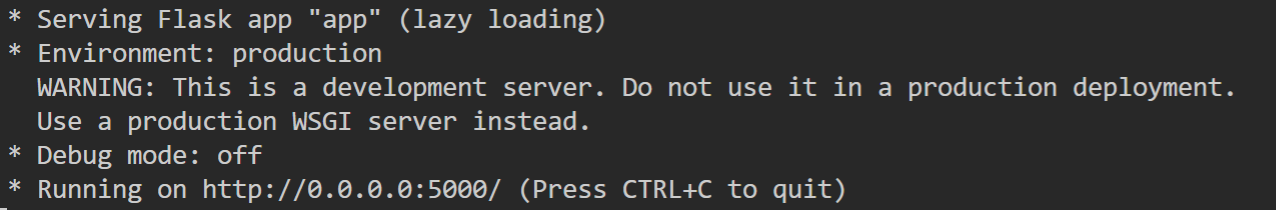
In your terminal, go to your project’s path by using the *cd* command



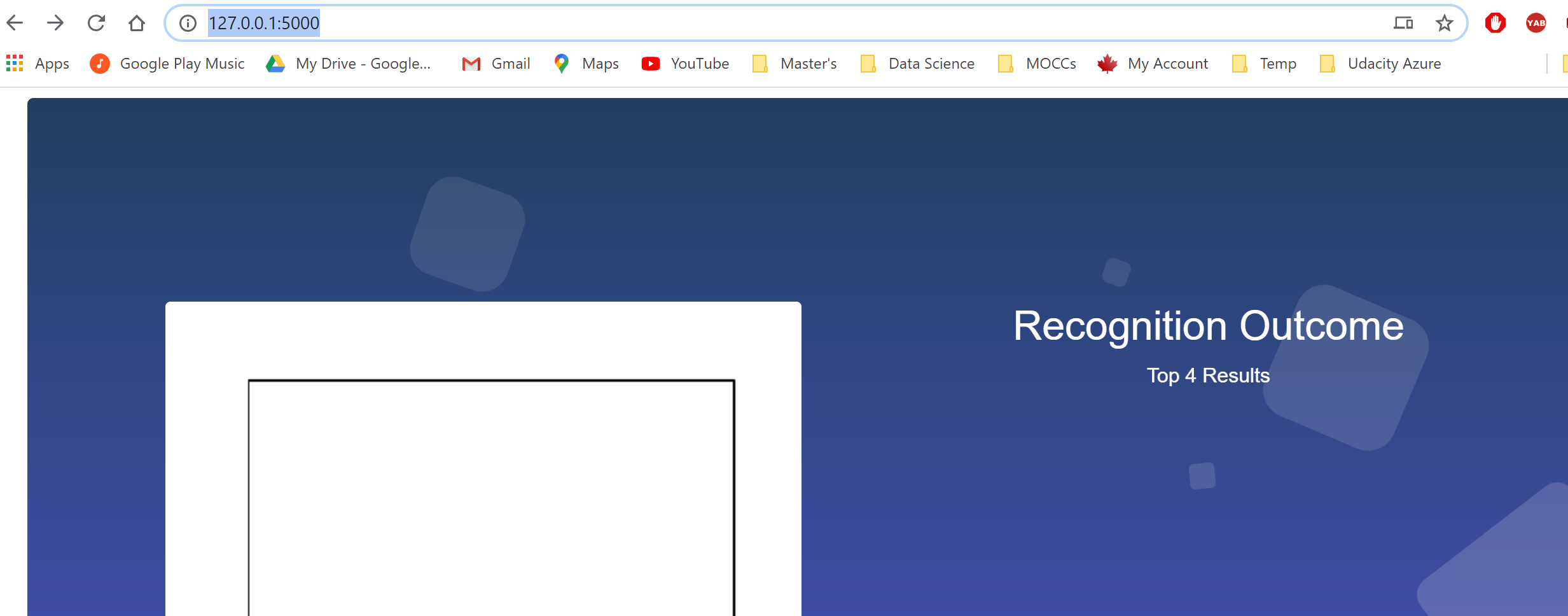
Finally, type *python app.py*. If you used a virtual environment, you need to activate it before executing that command.



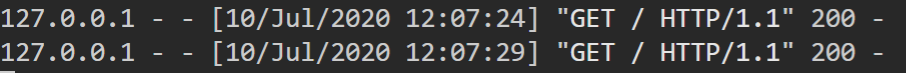
You will get an output similar to below image



Go to your web browser and type <http://127.0.0.1:5000/> (this is your local host) and voila!



As you interact with the classifier, you will notice these lines in your terminal. The number 200 informs that your GET requests have been successfully processed



Once you have finished, end the connection by typing *Ctrl + c* in your terminal