

<b>Team ID</b>	<b>PNT2022TMID50788</b>
<b>Project Name</b>	<b>AI-powered Nutrition Analyzer for FitnessEnthusiasts</b>

## TESTING

```
import numpy as np
from tensorflow.keras.models
import load_model
from tensorflow.keras.preprocessing import image
model=load_model('train.h5')
model=load_model('dataset.h5')
model=load_model('nutrition.h5')
img=image.load_img(r"/content/drive/MyDrive
Training/CNN/Dataset/TEST_SET/WATERMELON/3_100.jpg")
img
```



```
img=image.load_img(r"/content/drive/MyDrive
Training/CNN/Dataset/TEST_SET/WATERMELON/3_100.jpg",
target_size=(64,64))
img
```



```
x=image.img_to_array(img)
x
array([[[[255., 255., 255.],
[255., 255., 255.],
```

[255., 255., 255.],

...,

[255., 255., 255.],

[255., 255., 255.],

[255., 255., 255.]],

[[255., 255., 255.],

[255., 255., 255.],

[255., 255., 255.],

...,

[255., 255., 255.],

[255., 255., 255.],

[255., 255., 255.]],

[[255., 255., 255.],

[255., 255., 255.],

[255., 255., 255.],

...,

[255., 255., 255.],

[255., 255., 255.],

[255., 255., 255.]],

...,

[[255., 255., 255.],

[255., 255., 255.],

[255., 255., 255.],

...,

[255., 255., 255.],

[255., 255., 255.],

[255., 255., 255.]],

```
[[255., 255., 255.],  
 [255., 255., 255.],  
 [255., 255., 255.],  
 ...,  
 [255., 255., 255.],  
 [255., 255., 255.],  
 [255., 255., 255.]],
```

```
[[255., 255., 255.],  
 [255., 255., 255.],  
 [255., 255., 255.],  
 ...,  
 [255., 255., 255.],  
 [255., 255., 255.],  
 [255., 255., 255.]]], dtype=float32)
```

```
x=np.expand_dims(x,axis=0)
```

```
[[255., 255., 255.],  
 [255., 255., 255.],  
 [255., 255., 255.],  
 ...,  
 [255., 255., 255.],  
 [255., 255., 255.],  
 [255., 255., 255.]],
```

```
[[255., 255., 255.],  
 [255., 255., 255.],  
 [255., 255., 255.],  
 ...,  
 [255., 255., 255.],  
 [255., 255., 255.],  
 [255., 255., 255.]],
```

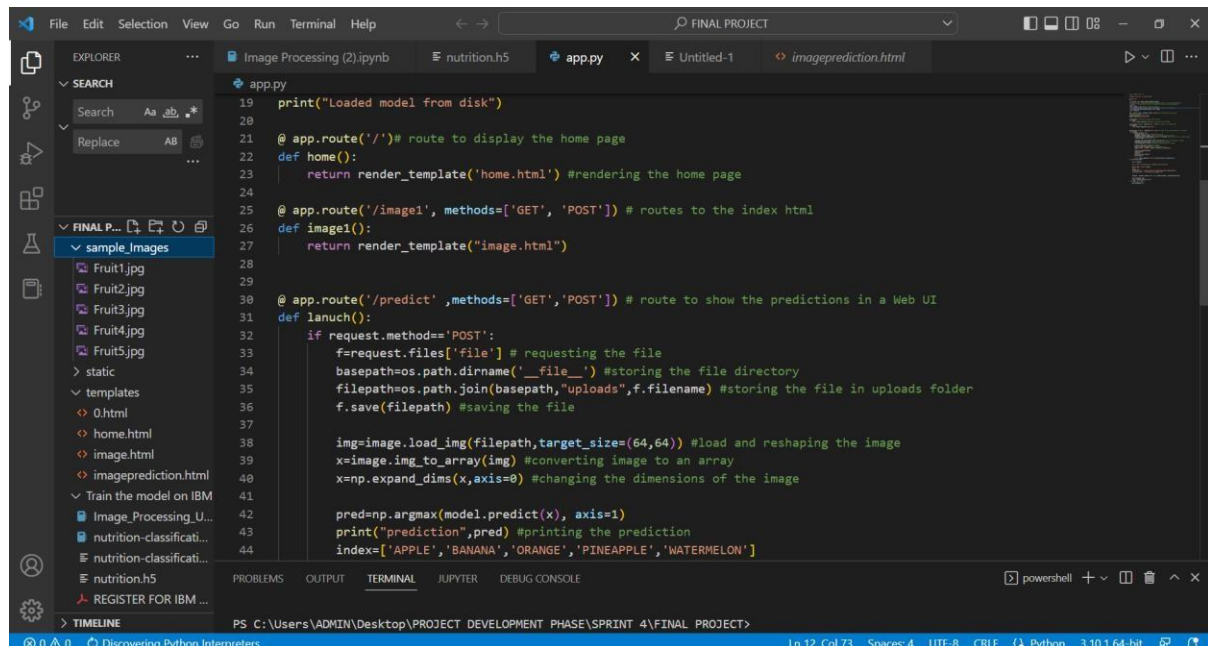
```
[[255., 255., 255.],  
 [255., 255., 255.],  
 [255., 255., 255.],  
 ...,
```

```

        [255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.] ]], dtype=float32)
pred = model.predict
pred
array
([[0.25227112, 0.17414774, 0.15219809, 0.20493415, 0.21644896],
 [0.26760292, 0.1759095 , 0.15206912, 0.19424875, 0.21016978],
 [0.26474723, 0.165203 , 0.14452063, 0.20434381, 0.2211853 ],
 ...,
 [0.24550524, 0.1721549 , 0.16282505, 0.21065485, 0.20885986],
 [0.25395462, 0.1735253 , 0.16055605, 0.20655352, 0.20541045],
 [0.24495909, 0.15889102, 0.16927534, 0.20705006, 0.21982446]],
 dtype=float32
<bound method Model.predict of <keras.engine.
sequential.Sequential object at 0x7f94abfd7c10>>
predict_x=model.predict(x_test)
classes_x=np.argmax(predict_x,axis=1)
classes_x
array([0, 0, 0, ..., 0, 0, 0])
x_test.class_indices
index=['APPLE','BANANA','ORANGE','WATERMELON','PINEAPPLE']
result=str(index[classes_x[0]])
result
'Watermelon'

```

## 8.1 TEST CASES

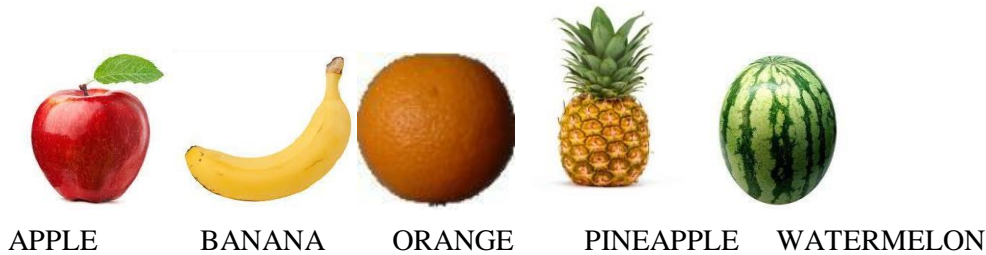


The screenshot shows a Visual Studio Code editor window titled 'FINAL PROJECT'. The Explorer sidebar on the left shows a file structure with a 'sample\_images' folder containing five fruit images (Fruit1.jpg to Fruit5.jpg) and a 'templates' folder with 'home.html', 'image.html', and 'imageprediction.html'. The main editor area displays the 'app.py' file, which is a Python Flask application. The code includes a print statement for the loaded model, routes for the home page, image upload, and prediction, and a launch function for the web UI. The prediction function uses a pre-trained model to classify images into categories like 'APPLE', 'BANANA', 'ORANGE', 'PINEAPPLE', and 'WATERMELON'. The status bar at the bottom indicates the file is at line 12, column 73, and the Python interpreter is 3.10.1 64-bit.

```
19 print("Loaded model from disk")
20
21 @ app.route('/')# route to display the home page
22 def home():
23     return render_template('home.html') #rendering the home page
24
25 @ app.route('/image1', methods=['GET', 'POST']) # routes to the index html
26 def image1():
27     return render_template("image.html")
28
29
30 @ app.route('/predict' ,methods=['GET','POST']) # route to show the predictions in a Web UI
31 def lanuch():
32     if request.method=='POST':
33         f=request.files['file'] # requesting the file
34         basepath=os.path.dirname('__file__') #storing the file directory
35         filepath=os.path.join(basepath,"uploads",f.filename) #storing the file in uploads folder
36         f.save(filepath) #saving the file
37
38         img=image.load_img(filepath,target_size=(64,64)) #load and reshaping the image
39         x=image.img_to_array(img) #converting image to an array
40         x=np.expand_dims(x,axis=0) #changing the dimensions of the image
41
42         pred=np.argmax(model.predict(x), axis=1)
43         print("prediction",pred) #printing the prediction
44         index=['APPLE','BANANA','ORANGE','PINEAPPLE','WATERMELON']
```

## 8.2 USER ACCEPTANCE TESTING

User Acceptance Testing (UAT) is a type of testing performed by the end user or the client to verify/accept the software system before moving the software application to the production environment. UAT is done in the final phase of testing after functional, integration and system testing is done. The main Purpose of UAT is to validate end to end business flow. It does not focus on cosmetic errors, spelling mistakes or system testing. User Acceptance Testing is carried out in a separate testing environment with production-like data setup. It is a kind of black box testing where two or more end-users will be involved. Need of User Acceptance Testing arises once software has undergone Unit, Integration and System testing because developers might have built software based on requirements document by their own understanding and further required changes during development may not be effectively communicated to them, so for testing whether the final product is accepted by client/end-user, user acceptance testing is needed.



## PERFORMANCE TESTING

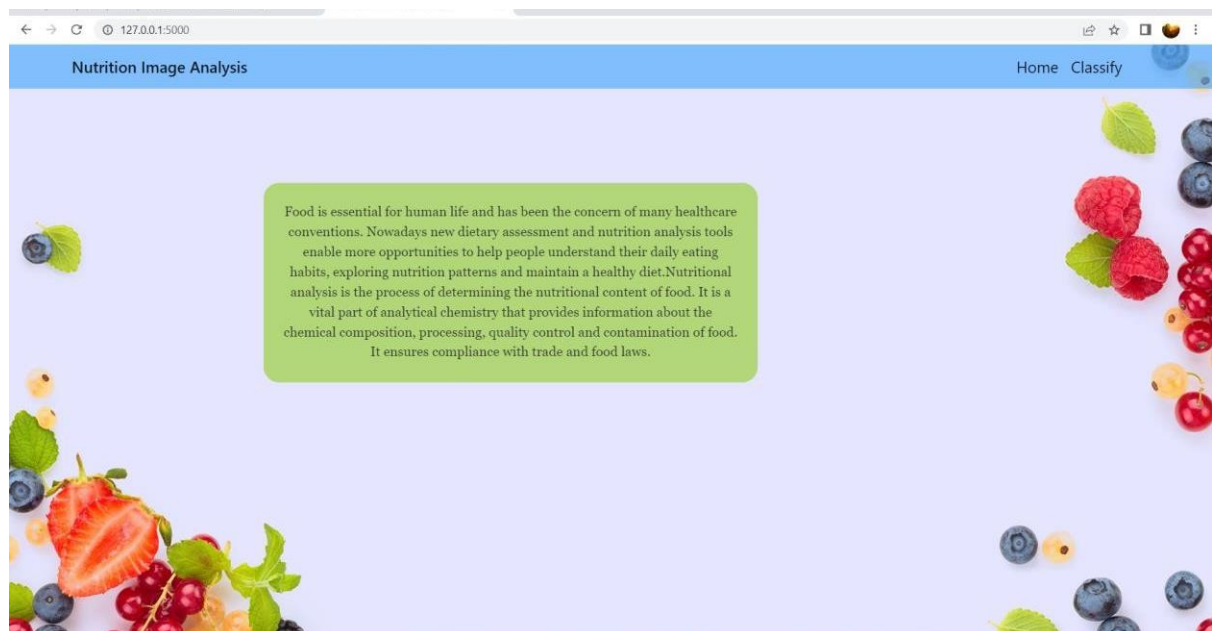
/tmp/wsuser/ipykernel\_165/2706448856.py:2: UserWarning: `Model.fit\_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.

```

classifier.fit_generator(
Epoch 1/20
828/828 [=====] - 56s 67ms/step - loss: 0.6839 - accuracy: 0.7255 -
val_loss: 0.4220 - val_accuracy: 0.8308
Epoch 2/20
828/828 [=====] - 54s 66ms/step - loss: 0.4299 - accuracy: 0.8366 -
val_loss: 0.3379 - val_accuracy: 0.8712
Epoch 3/20
828/828 [=====] - 55s 66ms/step - loss: 0.3900 - accuracy: 0.8523 -
val_loss: 0.3233 - val_accuracy: 0.8697
Epoch 4/20
828/828 [=====] - 55s 66ms/step - loss: 0.3700 - accuracy: 0.8596 -
val_loss: 0.3347 - val_accuracy: 0.8775
Epoch 5/20
828/828 [=====] - 55s 66ms/step - loss: 0.3381 - accuracy: 0.8748 -
val_loss: 0.3260 - val_accuracy: 0.8719
Epoch 6/20
828/828 [=====] - 54s 65ms/step - loss: 0.3316 - accuracy: 0.8731 -
val_loss: 0.2455 - val_accuracy: 0.9108
Epoch 7/20
828/828 [=====] - 55s 66ms/step - loss: 0.3080 - accuracy: 0.8821 -
val_loss: 0.2553 - val_accuracy: 0.9072
Epoch 8/20
828/828 [=====] - 54s 65ms/step - loss: 0.2942 - accuracy: 0.8891 -
val_loss: 0.2722 - val_accuracy: 0.8990
Epoch 9/20
828/828 [=====] - 54s 66ms/step - loss: 0.2709 - accuracy: 0.8983 -
val_loss: 0.2202 - val_accuracy: 0.9176
Epoch 10/20

```

828/828 [=====] - 54s 65ms/step - loss: 0.2648 - accuracy: 0.8949 -  
val\_loss: 0.2655 - val\_accuracy: 0.8946  
Epoch 11/20  
828/828 [=====] - 54s 66ms/step - loss: 0.2417 - accuracy: 0.9101 -  
val\_loss: 0.2126 - val\_accuracy: 0.9174  
Epoch 12/20  
828/828 [=====] - 53s 65ms/step - loss: 0.2282 - accuracy: 0.9130 -  
val\_loss: 0.2247 - val\_accuracy: 0.9108  
Epoch 13/20  
828/828 [=====] - 52s 63ms/step - loss: 0.2246 - accuracy: 0.9132 -  
val\_loss: 0.2408 - val\_accuracy: 0.9070  
Epoch 14/20  
828/828 [=====] - 54s 65ms/step - loss: 0.2162 - accuracy: 0.9176 -  
val\_loss: 0.1503 - val\_accuracy: 0.9454  
Epoch 15/20  
828/828 [=====] - 54s 65ms/step - loss: 0.1903 - accuracy: 0.9261 -  
val\_loss: 0.1458 - val\_accuracy: 0.9420  
Epoch 16/20  
828/828 [=====] - 54s 65ms/step - loss: 0.1776 - accuracy: 0.9316 -  
val\_loss: 0.1147 - val\_accuracy: 0.9526  
Epoch 17/20  
828/828 [=====] - 54s 65ms/step - loss: 0.1696 - accuracy: 0.9357 -  
val\_loss: 0.1248 - val\_accuracy: 0.9553  
Epoch 18/20  
828/828 [=====] - 54s 65ms/step - loss: 0.1477 - accuracy: 0.9442 -  
val\_loss: 0.1842 - val\_accuracy: 0.9321  
Epoch 19/20  
349/828 [=====> ..... ] - ETA: 24s - loss: 0.1653 - accuracy: 0.9323







← → ↻ 127.0.0.1:5000/image1


🔍 ⭐ 🗨 🌐

Nutrition Image Analysis

Home Classify

Upload image to classify

Choose



Food Classified is:

BANANA

[('sugar\_g': 12.3, 'fiber\_g': 2.6, 'serving\_size\_g': 100.0, 'sodium\_mg': 1, 'name': 'banana', 'potassium\_mg': 22, 'fat\_saturated\_g': 0.1, 'fat\_total\_g': 0.3, 'calories': 89.4, 'cholesterol\_mg': 0, 'protein\_g': 1.1, 'carbohydrates\_total\_g': 23.2)]

