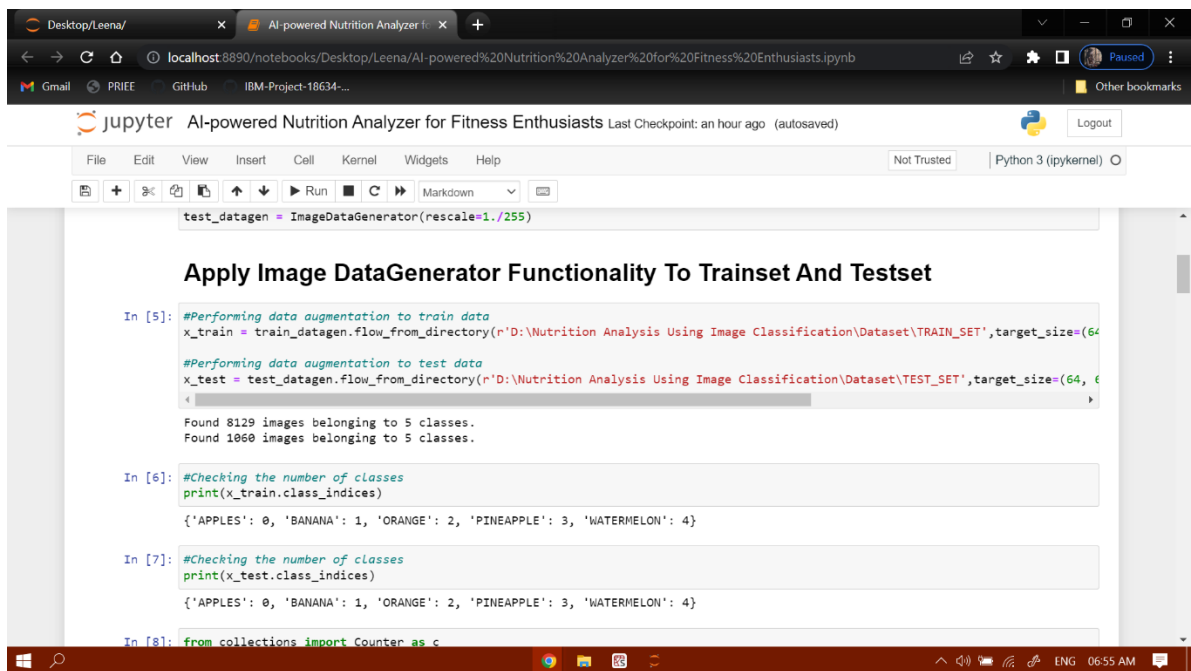


## Acceptance Testing UAT Execution & Report Submission

Date	19 November 2022
Team ID	PNT2022TMID12370
Project Name	Project - AI-Powered Nutrition Analyzer For Fitness Enthusiasts
Maximum Marks	4 Marks

### 1. Dataset Loaded successfully:



```
test_datagen = ImageDataGenerator(rescale=1./255)

#Performing data augmentation to train data
x_train = train_datagen.flow_from_directory(r'D:\Nutrition Analysis Using Image Classification\Dataset\TRAIN_SET',target_size=(64, 64))

#Performing data augmentation to test data
x_test = test_datagen.flow_from_directory(r'D:\Nutrition Analysis Using Image Classification\Dataset\TEST_SET',target_size=(64, 64))

Found 8129 images belonging to 5 classes.
Found 1060 images belonging to 5 classes.

#Checking the number of classes
print(x_train.class_indices)

{'APPLES': 0, 'BANANA': 1, 'ORANGE': 2, 'PINEAPPLE': 3, 'WATERMELON': 4}

#Checking the number of classes
print(x_test.class_indices)

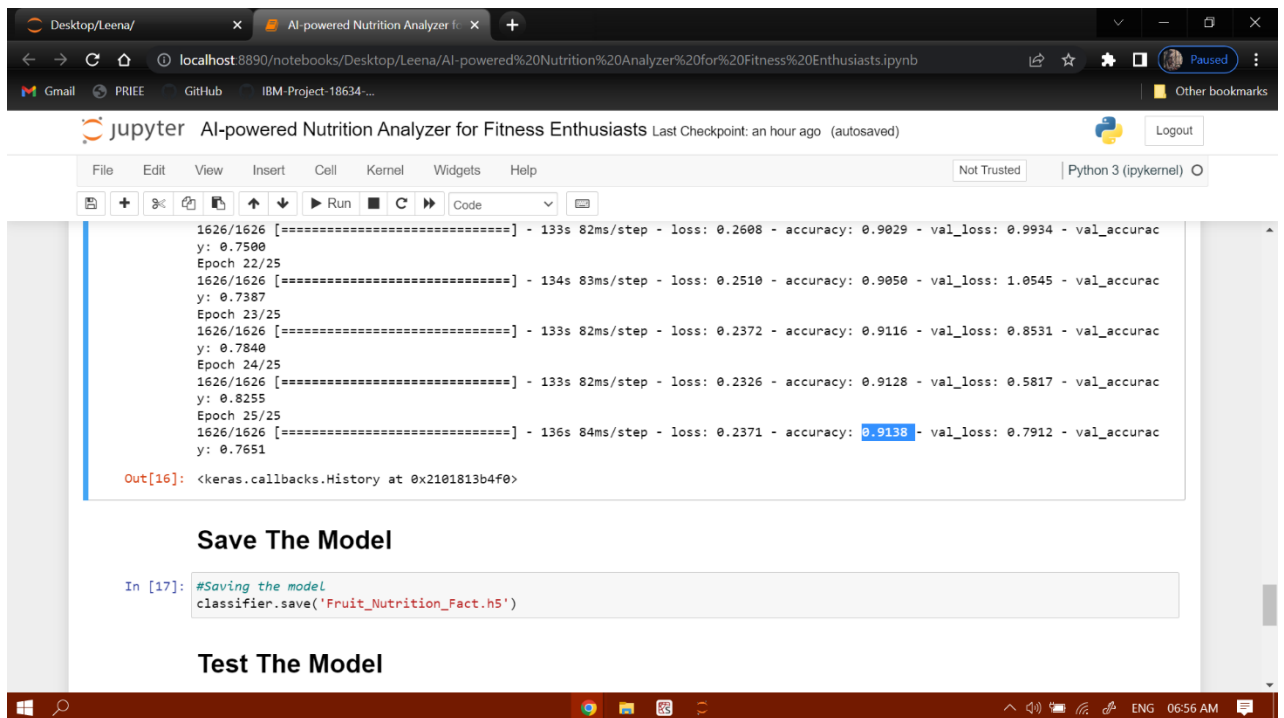
{'APPLES': 0, 'BANANA': 1, 'ORANGE': 2, 'PINEAPPLE': 3, 'WATERMELON': 4}

from collections import Counter as c
```

Testcase: Found both training and test dataset

Result: Pass

## 2. Accuracy of Model in identifying fruits:



The screenshot shows a Jupyter Notebook titled "AI-powered Nutrition Analyzer for Fitness Enthusiasts". The notebook is running on a local host. The output of the training process is displayed, showing the progress of the model over 25 epochs. The accuracy of the model is 0.9138, which is highlighted in blue. The notebook also shows the code to save the model as "Fruit\_Nutrition\_Fact.h5".

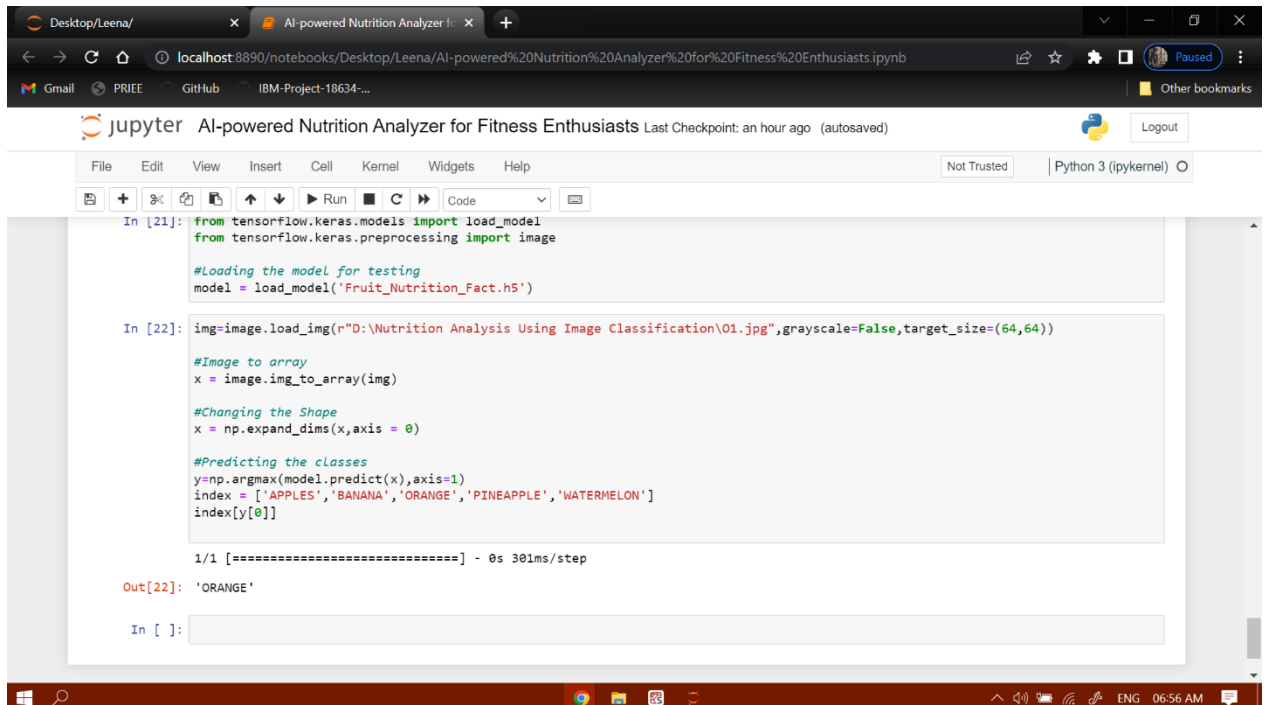
```
1626/1626 [=====] - 133s 82ms/step - loss: 0.2608 - accuracy: 0.9029 - val_loss: 0.9934 - val_accuac
y: 0.7500
Epoch 22/25
1626/1626 [=====] - 134s 83ms/step - loss: 0.2510 - accuracy: 0.9050 - val_loss: 1.0545 - val_accuac
y: 0.7387
Epoch 23/25
1626/1626 [=====] - 133s 82ms/step - loss: 0.2372 - accuracy: 0.9116 - val_loss: 0.8531 - val_accuac
y: 0.7840
Epoch 24/25
1626/1626 [=====] - 133s 82ms/step - loss: 0.2326 - accuracy: 0.9128 - val_loss: 0.5817 - val_accuac
y: 0.8255
Epoch 25/25
1626/1626 [=====] - 136s 84ms/step - loss: 0.2371 - accuracy: 0.9138 - val_loss: 0.7912 - val_accuac
y: 0.7651

Out[16]: <keras.callbacks.History at 0x2101813b4f0>
```

**Save The Model**

```
In [17]: #Saving the model
classifier.save('Fruit_Nutrition_Fact.h5')
```

**Test The Model**



The screenshot shows the same Jupyter Notebook with the code to load the model and test it. The model is loaded from "Fruit\_Nutrition\_Fact.h5". The test image is loaded from "D:\Nutrition Analysis Using Image Classification\01.jpg". The model predicts the class "ORANGE".

```
In [21]: from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image

#Loading the model for testing
model = load_model('Fruit_Nutrition_Fact.h5')

In [22]: img=image.load_img(r"D:\Nutrition Analysis Using Image Classification\01.jpg",grayscale=False,target_size=(64,64))

#Image to array
x = image.img_to_array(img)

#Changing the Shape
x = np.expand_dims(x,axis = 0)

#Predicting the classes
y=np.argmax(model.predict(x),axis=1)
index = ['APPLES','BANANA','ORANGE','PINEAPPLE','WATERMELON']
index[y[0]]

1/1 [=====] - 0s 301ms/step

Out[22]: 'ORANGE'

In [ ]:
```

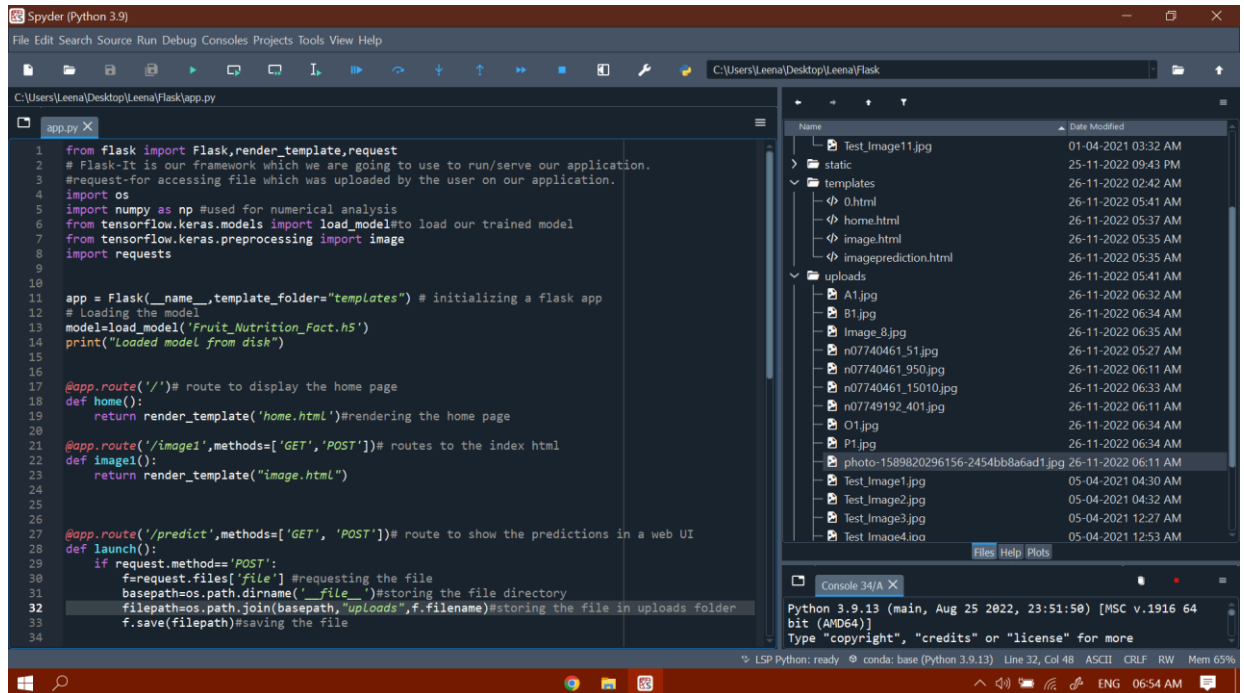
Testcase: Accuracy And Image Classification?

Result: Pass

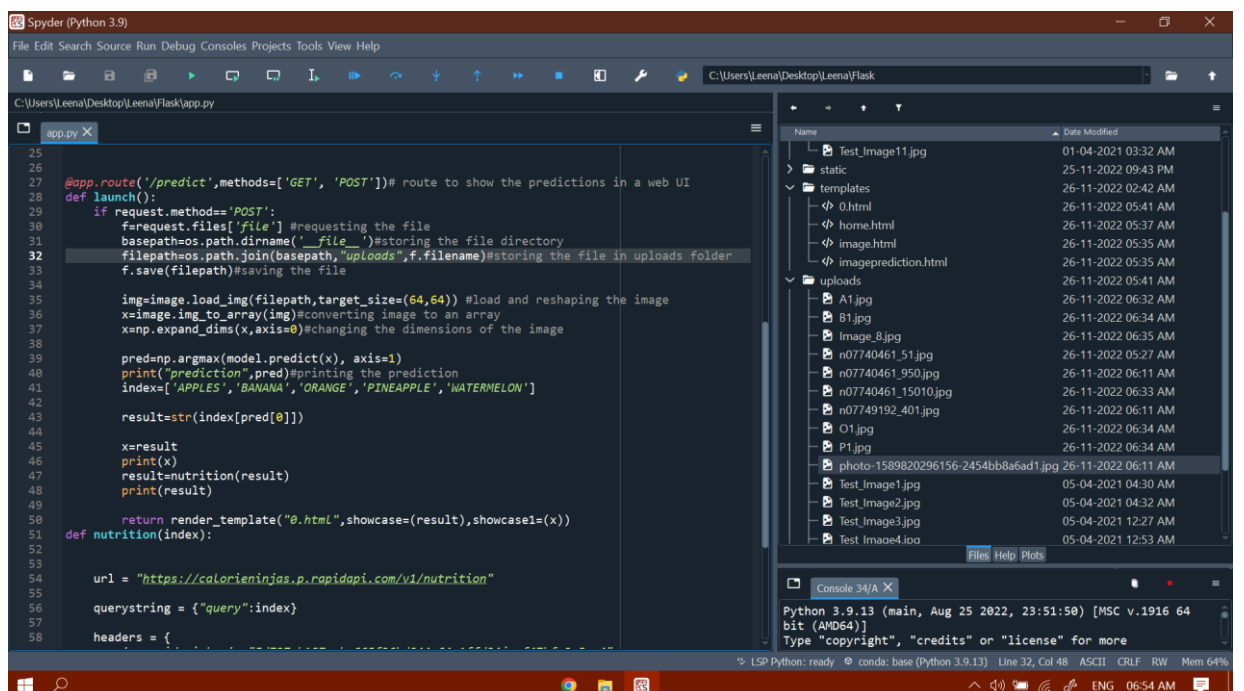
Accuracy Achieved: 91%

Image Classification: Success

### 3. Build Python Code:



```
1 from flask import Flask,render_template,request
2 # Flask-It is our framework which we are going to use to run/serve our application.
3 #request-for accessing file which was uploaded by the user on our application.
4 import os
5 import numpy as np #used for numerical analysis
6 from tensorflow.keras.models import load_model#to load our trained model
7 from tensorflow.keras.preprocessing import image
8 import requests
9
10
11 app = Flask(__name__,template_folder="templates") # initializing a flask app
12 # Loading the model
13 model=load_model('Fruit_Nutrition_Fact.h5')
14 print("Loaded model from disk")
15
16
17 @app.route('/')# route to display the home page
18 def home():
19     return render_template('home.html')#rendering the home page
20
21 @app.route('/image1',methods=['GET','POST'])# routes to the index html
22 def image1():
23     return render_template("image.html")
24
25
26
27 @app.route('/predict',methods=['GET', 'POST'])# route to show the predictions in a web UI
28 def launch():
29     if request.method=='POST':
30         f=request.files['file'] #requesting the file
31         basepath=os.path.dirname(__file__)#storing the file directory
32         filepath=os.path.join(basepath,"uploads",f.filename)#storing the file in uploads folder
33         f.save(filepath)#saving the file
34
```

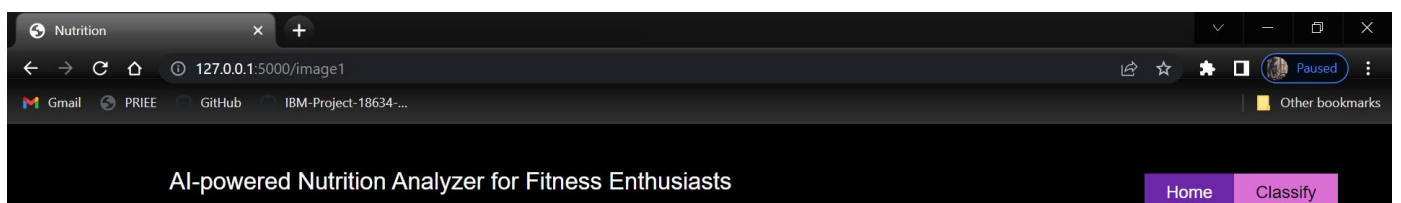
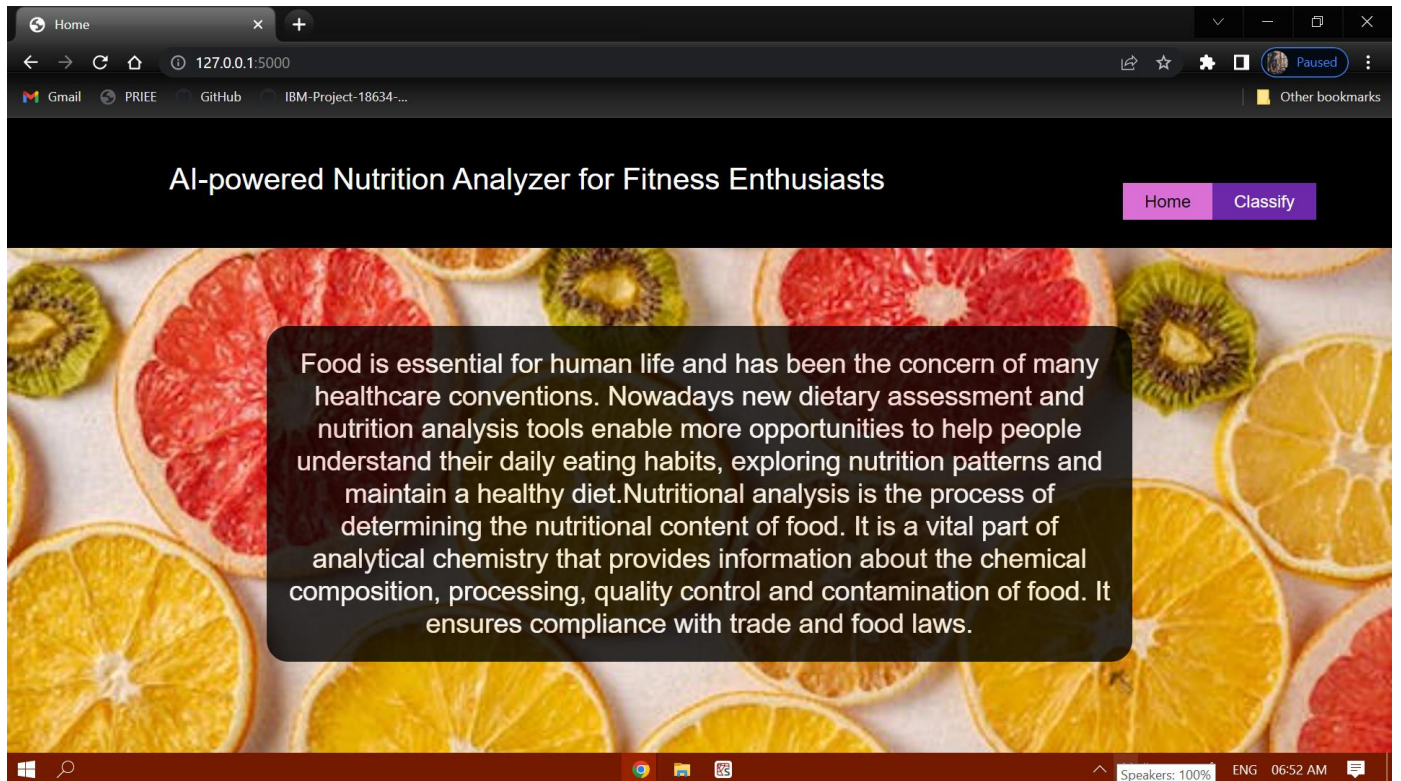


```
25
26
27 @app.route('/predict',methods=['GET', 'POST'])# route to show the predictions in a web UI
28 def launch():
29     if request.method=='POST':
30         f=request.files['file'] #requesting the file
31         basepath=os.path.dirname(__file__)#storing the file directory
32         filepath=os.path.join(basepath,"uploads",f.filename)#storing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

Testcase: Run the application

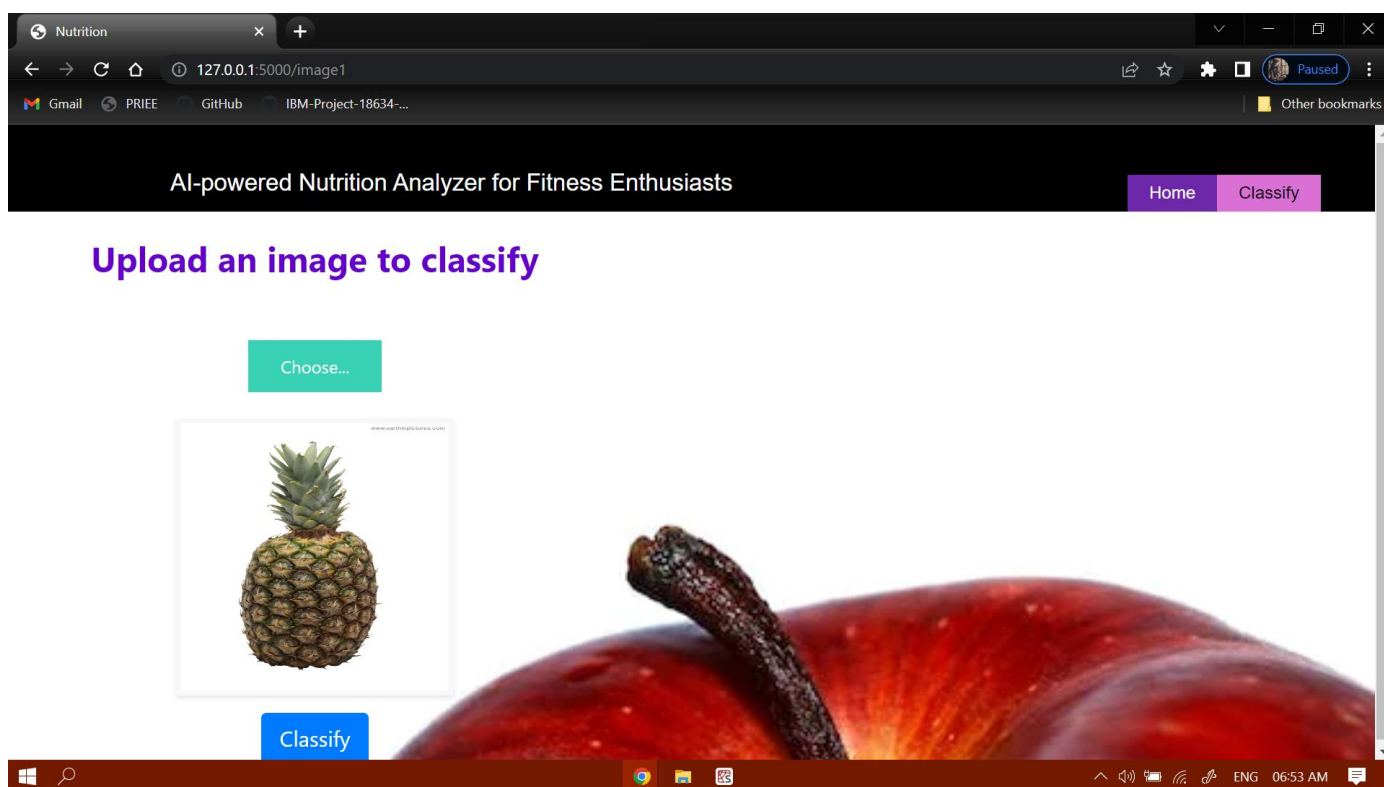
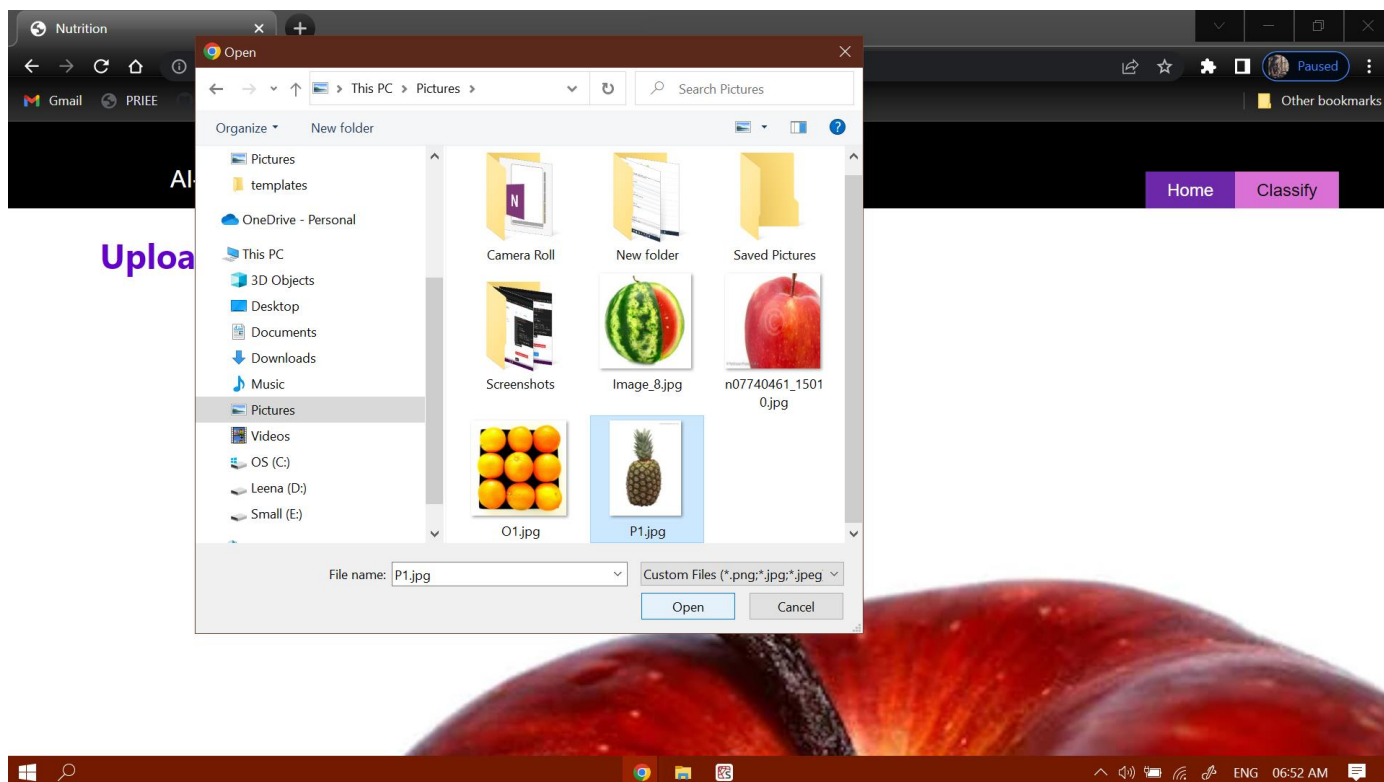
Result: Pass

#### 4. Final Output:

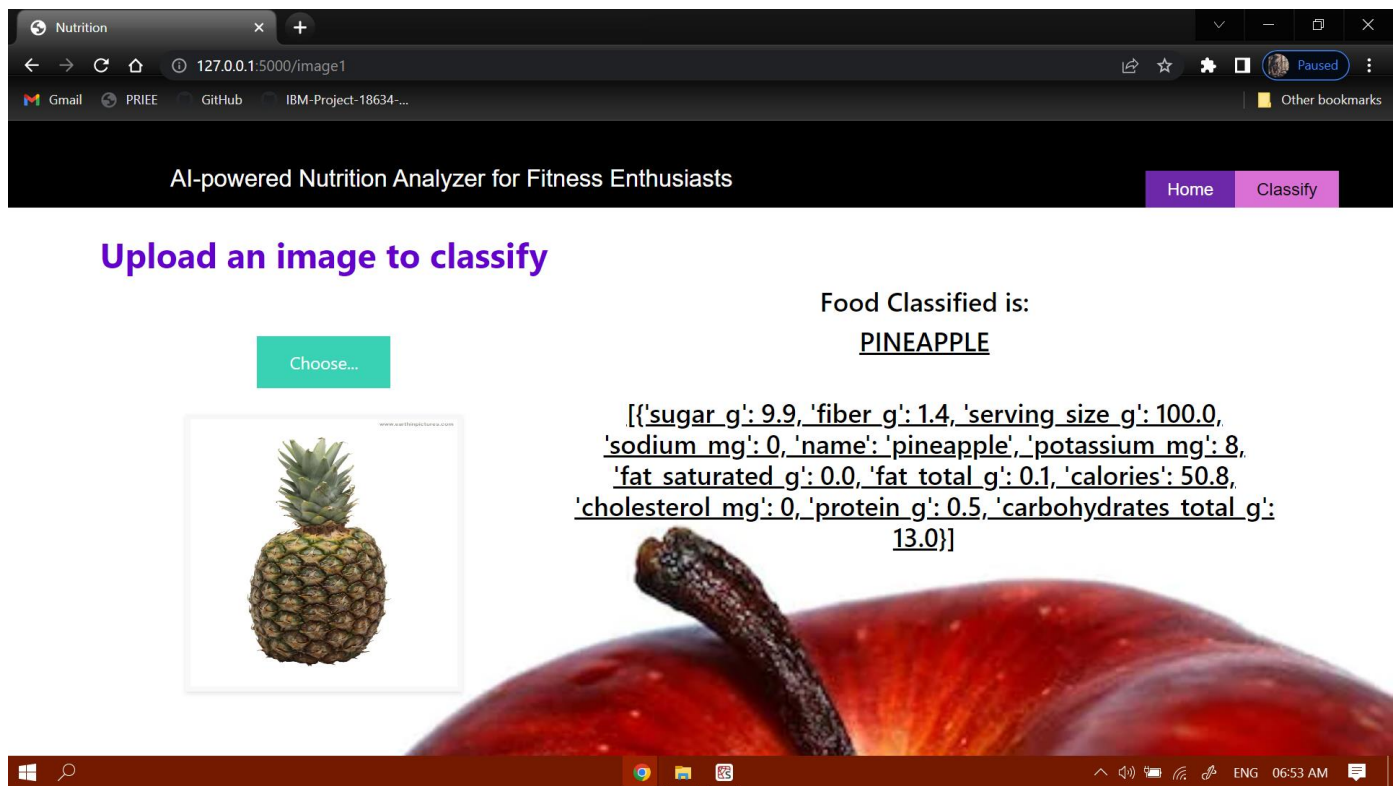


**Upload an image to classify**









Testcase: Identifies fruit correctly when classify button is clicked

Result: Pass