
ARTIFICIALLY INTELLIGENT LOGISTIC MANAGEMENT SYSTEM

SUBMITTED TO

PROFESSOR: MARCOS BITTENCOURT

BY – GROUP 4

DHRUVA LOYA

NICHOLAS WILLIAMS

PRASAD BABU VENUGOPAL

DEVENDRANATH SANTHOSH CHERUKURI

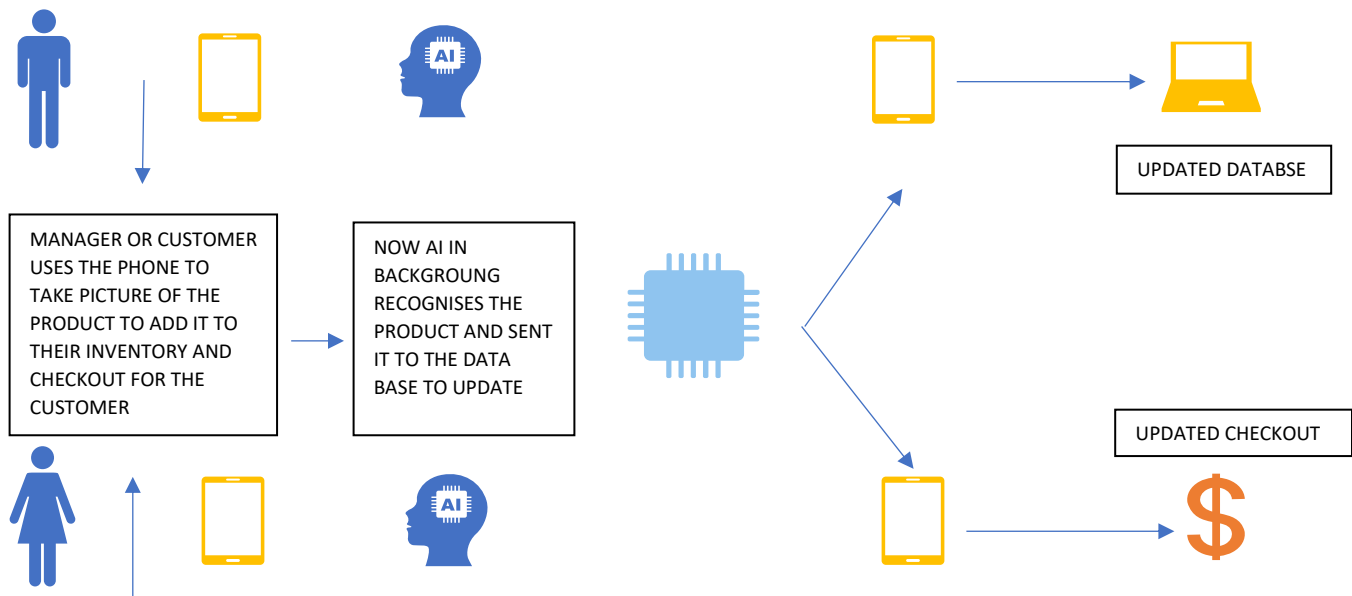
INDEX

PROJECT DESCRIPTION	2
VISUAL REPRESENTATION	
BUSINESS BENEFIT	
ARTIFACT	3
CONTEXT	
IMPACT	3
KPI	4
IMAGE RECOGNITION ACCURACY	
INVENTORY ACCURACY	
DATA ASSUMPTIONS	4
DATA LIMITATIONS	4
MODE ARCHITECTURE& S/W PIPELINE	5
MODEL STRUCTURE	6
SOFTWARE TOOLS	6
IMAGE ECOGNITION	7
PROJECT OUTCOMES	13
ALGORITHM	13
MOBILE APP	17
INVENTORY MANAGEMENT	21
REFERENCES	31

PROJECT NAME: ARTIFICIALLY INTELLIGENT LOGISTIC MANAGEMENT SYSTEM

PROJECT DESCRIPTION: THIS PRODUCT WORKS IN TWO WAYS (STORE OWNERS AND CUSTOMERS), THIS PROJECTS GOAL IS TO HAVE A MOBILE PHONE-BASED IMAGE CLASSIFICATION USING DEEP LEARNING. STORE MANAGERS USES THIS APPLICATION TO UPDATE THEIR STOCK INVENTORY BY TAKING A PICTURE EVERYTIME THEY WANT TO UPDATE A NEW PRODUCT, WHEN THEY TOOK THE PICTURE FROM OUR APP THE AI BEHIND THE APP WILL PROCESS THE IMAGE AND ADDS IN TO THE CLOUD BASED STRORE INVENTORY DATABASE, SIMULTANEOUSLY WHEN CUSTOMERS PICKING THEIR PRODUCTS THEY WILL TAKE A PICTURE AND THE AI BEHIND THE APP IS GOING TO RECOGNISE THE PRODUCT AND ADD THE PRODUCT TO THE CART AND ALSO REMOVES THE SAME PRODUCT FROM THE STORE'S PRODUCT INVENOTRY DATABASE AT THE SAME TIME. THIS GIVES THE BENEFIT OF EASY INVENTORY MANAGEMENT TO STORE OWNERS AS WELL AS EASY CHECKOUT FOR THE CUSTOMERS.

VISUAL REPRESENTATION:



WHERE IS AI IN THE SOLUTION

THERE IS IMAGE CLASSIFICATION USING DEEP LEARNING. ONCE A IMAGE IS CAPTURED, ALGORITHM WILL CLASSIFY THE OBJECT AND UPDATE THE USER CART WITH QUANTITY BASED PRICE.

WHAT IS BUSINESS BENEFIT

THE QUANTITY GETS UPDATED IN OVERALL STOCK DATABASE OF BUSINESS OWNER. SO THE BUSINESS OWNER WILL TAKE CARE OF LOGISTICS WITH EASE.

ARTIFACT: IMAGE CLASSIFICATION FOR GROCERY STORE

THIS PROJECT WAS DESIGNED TO SERVE THE CUSTOMERS AS WELL AS MANAGEMENT OF GROCERY STORE TO FASTER TRANSACTIONS AND AS WELL AS FASTER INVENTORY MANAGEMENT.

CONTEXT:

THIS PROJECTS GOAL IS TO HAVE A MOBILE PHONE-BASED IMAGE CLASSIFICATION USING DEEP LEARNING THIS ALLOWS EMPLOYEES TO TAKE PICTURES OF PRODUCT IN THE STORE WHENEVER THEY WANT TO STOCK IT UP AND THE APP IN BACKGROUND UPDATE THE CLOUD DATABASE INVENTORY. THEN WHEN CUSTOMERS ARE CHECKING OUT, THEY ALSO TAKE ANOTHER PICTURE THAT REMOVES STOCK FROM THIS SAME DATABASE AND SIMULTANEOUSLY THE CUSTOMER'S CART WILL BE UPDATED AUTOMATICALLY AND MAKES IT EASIER FOR THE CUSTOMER TO CHECKOUT. THIS GIVES THE BENEFIT OF EASY INVENTORY MANAGEMENT TO STORE OWNERS AND MANAGERS AS WELL AS CUSTOMERS

IMPACT:

THIS PROJECT WOULD HAVE A GREAT IMPACT IN THE CURRENT MARKET, FOR EXAMPLE AS WHEN WE SEE IN A TYPICAL WALMART STORE EMPLOYEES AND CUSTOMERS THEY SPEND A LOT OF TIME ON UPDATING THE INVENTORY AND CUSTOMERS CHECKING BARCODES FOR PRICES AND WAITING IN THE QUEUE FOR A LONG TIME TO CHECKOUT AGAIN BY THE EMPLOYEES. SO OUR SYSTEM MAKES IT EASIER, FAST, RELIABLE TO THE CUSTOMERS AS WELL AS EMPLOYEES

ALIGNMENT:

FOR THIS PROJECT WE HAVE A SERIES OF STEPS TO CREATE AND EXECUTE

FIRST, WE ARE GOING TO CREATE A MOBILE APP (IOS OR ANDROID) AND WEBPAGE WITH INVENTORY SIMULTANEOUSLY BY OUR TEAM MEMBERS. THEN WE ARE GOING TO WRITE THE MACHINE LEARNING ALGORITHM FOR IMAGE CLASSIFICATION WHICH WE ARE GOING TO INTEGRATE IT WITH THE APP LATER. NOW WE COME INTO THE TESTING PHASE WHERE WE TEST IT SEVERAL TIME TILL WE MAKE SURE THAT EVERYTHING PROPERLY. NOW WE ARE GOING TO POOL ALL THE RESOURCES THAT WE CREATED, GATHERED AND ADOPTED THEN WE ARE GOING TO EXECUTE OUR BRAINCHILD.

KPI:**1. ACCURACY OF IMAGE RECOGNITION**

THE IMAGE RECOGNITION MUST BE HIGHLY ACCURATE SO THAT THE INTERCEPTED OBJECT TO MATCH WITH ACTUAL OBJECT SCANNED. THIS CAN BE MEASURED WITH METRICS LIKE F1 SCORE AND ACCURACY SCORE.

2. INVENTORY ACCURACY

THE BUSINESS OWNER MUST BE ABLE TO MANAGE THE INVENTORY WITH THE UPDATED STOCK INFORMATION. THE ACCURACY OF THIS IS INFORMATION IS MEASURED WITH PERFORMANCE METRICS LIKE ACCURACY SCORE.

IDENTIFYING THE DATA REQUIREMENTS:

AS OUR GROUP PROJECT IS IMAGE CLASSIFICATION AND DIRECTLY RELATED TO GROCERY STORES, WE ARE GOING TO USE VARIOUS KINDS OF DATA.

OUR PRIMARY DATA REQUIREMENTS ARE DATASETS OF THE AVERAGE GROCERY PRODUCTS, IMAGES OF VARIOUS FOODS, UPDATED RULES AND PROTOCOLS FROM THE LOCAL GOVERNMENT.

RELATED DATA SOURCES:

[HTTPS://WWW.EPFL.CH/LABS/MMSPG/DOWNLOADS/FOOD-IMAGE-DATASETS/](https://www.epfl.ch/labs/mmSPG/downloads/food-image-datasets/)

[HTTPS://WWW.KAGGLE.COM/MOLTEAN/FRUITS](https://www.kaggle.com/moltean/fruits)

[HTTPS://GITHUB.COM/MARCUSKLASSON/GROCERYSTOREDATASET](https://github.com/marcusklason/grocerystoredataset)

[HTTP://AISDATASETS.INFORMATIK.UNI-FREIBURG.DE/FREIBURG GROCERIES DATASET/](http://aisdatasets.informatik.uni-freiburg.de/freiburg_groceries_dataset/)

DATA ASSUMPTIONS:

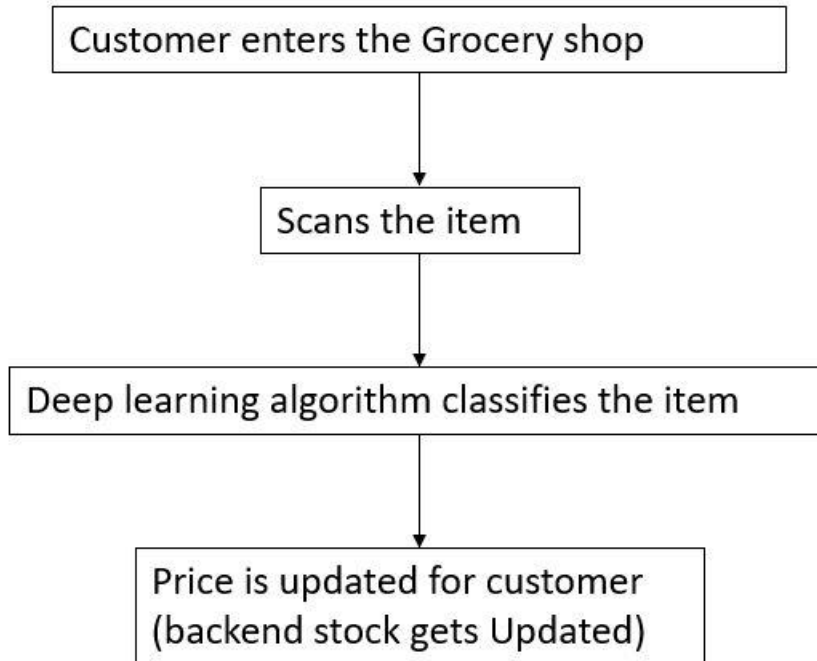
AS YOU CAN SEE FROM THE ABOVE-MENTIONED DATA SOURCES, WE ASSUME THAT THESE DATASETS ARE QUIET ENOUGH FOR OUR PROJECT AND THE DATA REGARDING THE GROCERY STORES PROVIDES US WITH ENOUGH INFORMATION THROUGHOUT OUR PROJECT.

LIMITATIONS & CONSTRAINTS:

ALTHOUGH WE HAVE MORE THAN REQUIRED INFORMATION IN THE ABOVE-MENTIONED DATASETS, WE ARE GOING TO LIMIT THE DATA THAT WE ARE GOING TO USE IN OUR PROJECT SO THAT WE CAN COMPLETE IT IN

GIVEN TIME, LIMITATIONS SUCH AS KEEPING THE DATABASE IN 5 TO 10 ROWS AND COLUMNS, USING LIMITED IMAGES OF FOOD, ALSO LIMITING THE APP TO FEWER FEATURES.

Capstone II – AI enabled Logistics System - Process flow



MODEL ARCHITECTURE AND SOFTWARE PIPELINE:

BECAUSE OUR GROUP PROJECT IS THE CLASSIFICATION OF IMAGES AND IS DIRECTLY RELATED TO GROCERY STORES, WE WILL BE USING VARIOUS DATA FORMS. FIRST DATA WILL BE PREPROCESSED AND THEN IT WILL BE SEGMENTED INTO TWO PARTS CALLED TEST DATA AND TRAIN DATA. AS THIS WILL REDUCE THE CHANCES

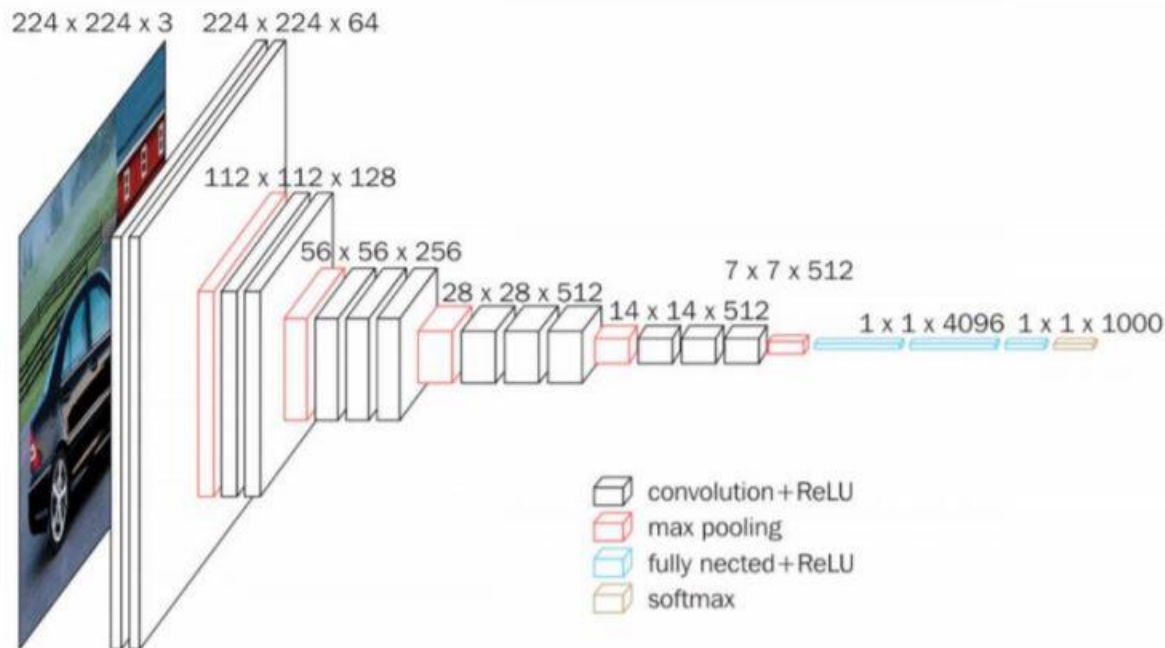
OF GETTING FATAL ERRORS. THIS IS THE PIPELINE FOR THE FLOW OF DATA IN OUR MODEL.

**ASSUMPTIONS, LIMITATION & CONSTRAINTS:**

AS YOU CAN SEE FROM THE ABOVE-MENTIONED DOCUMENTS PROVIDED, WE ASSUME THAT THE DATASET WE HAVE IS QUIET ENOUGH FOR OUR PROJECT AS CURRENTLY OUR PROJECT IS IN THE MODELLING PHASE OF THE DEVELOPMENT. BUT ONCE THE MODEL IS DEVELOPED, WE CAN ADD MORE DATA IN THE DATABASE. AS OF NOW WE WILL RESTRICT THE DATA TO 25 ROWS AND COLUMNS IN DATABASE SO THAT WE CAN ACCOMPLISH THE TASK IN THE GIVEN PERIOD. IN THE SAME WAY BE WILL BE RESTRICTING OUR APPLICATION TO FEWER FEATURES.

MODEL STRUCTURE:

STRUCTURE OF OUR MODEL WILL CONSIST OF DIFFERENT LAYERS LIKE DATA LABELLING, DATA PREPROCESSING, TESTING AND TRAINING AND OUTPUT. FOR CLASSIFICATION OF THE IMAGE WE WILL MAKE USE OF VGG-16 MODEL, WHERE WE WILL PREPROCESS OUR DATA, TRAIN AND TEST OUR DATA. THIS IS THE ARCHITECTURE OF OUR IMAGE PROCESSING MODEL.



SOFTWARE TOOLS:

FOR THIS PROJECT WE WILL BE USING JUPYTER NOTEBOOK AS OUR IDE. MYSQL WILL BE USED AS OUR DATABASE. ANDROID SDK MANAGER WILL BE USED TO VISUALIZE THE FRONT END OF THE MODEL. VISUAL STUDIO WILL BE USED TO CODE THE FRONT END. LABEL IMAGER WAS USED TO LABEL THE IMAGES IN THE DATABASE.

CALIBRATION OF MODEL TECHNIQUES:

WE'LL APPLY VARIOUS TYPES OF MACHINE LEARNING ALGORITHMS AND TEST OUR MODEL'S ACCURACY. WE WILL TRY TO IMPROVE ACCURACY OF OUR MODEL BY ADDING PARAMETERS. WE WILL FIND OUT WHICH FEATURES ARE THE MOST IMPACTING ON THE ACCURACY OF THE MODEL. WE WILL USE THE TECHNIQUE LIKE PERMUTATION FEATURE IMPORTANCE WHICH CAN HELP US TO IDENTIFY SALIENT FEATURES IN DATA WHICH WILL HELP IN EXPLAINING THE MODEL. WE WILL USE THE OUTPUT OF A FEATURE SELECTION METHOD TO REDUCE THE NUMBER OF NOISY FEATURES GOING INTO THE TRAINING PROCESS.

PROTOTYPE:

FOR OUR PROJECT WE USED IMAGE DETECTION ALGORITHM TO EXECUTE OUR PROJECT USING DARKNET AND YOLO3 WEIGHTS

IMAGE RECOGNITION:

IMAGE OR OBJECT DETECTION IS A COMPUTER TECHNOLOGY THAT PROCESSES THE IMAGE AND DETECTS OBJECTS IN IT. PEOPLE OFTEN CONFUSE IMAGE DETECTION WITH IMAGE CLASSIFICATION. ALTHOUGH THE DIFFERENCE IS RATHER CLEAR. IF YOU NEED TO CLASSIFY IMAGE ITEMS, YOU USE CLASSIFICATION. BUT IF YOU JUST NEED TO LOCATE THEM, FOR EXAMPLE, FIND OUT THE NUMBER OF OBJECTS IN THE PICTURE, YOU SHOULD USE IMAGE DETECTION.

YOLOV3:

PRIOR DETECTION SYSTEMS REPURPOSE CLASSIFIERS OR LOCALIZERS TO PERFORM DETECTION. THEY APPLY THE MODEL TO AN IMAGE AT MULTIPLE LOCATIONS AND SCALES. HIGH SCORING REGIONS OF THE IMAGE ARE CONSIDERED DETECTIONS.

WE USE A TOTALLY DIFFERENT APPROACH. WE APPLY A SINGLE NEURAL NETWORK TO THE FULL IMAGE. THIS NETWORK DIVIDES THE IMAGE INTO REGIONS AND PREDICTS BOUNDING BOXES AND PROBABILITIES FOR EACH REGION. THESE BOUNDING BOXES ARE WEIGHTED BY THE PREDICTED PROBABILITIES.

THIS MAKES IT EXTREMELY FAST, MORE THAN 1000X FASTER THAN R-CNN AND 100X FASTER THAN FAST R-CNN. SEE OUR PAPER FOR MORE DETAILS ON THE FULL SYSTEM.

BELOW GIVEN IS THE IMAGE OF OUR PROJECT PROTOTYPE IN WHICH YOU CAN SEE THAT WE ARE FEEDING THE DATA AND THE OUTPUT WHERE YOU CAN SEE THE ACCURACY THAT WE ACHIEVED IN DETECTING THE FRUITS "APPLE: AND "BANANA". WHERE WE ACHIEVED 91.75% ACCURACY WHILE DETECTING BANANA AND APPLE 97.4%. IT IS NOT A 100 PERCENT ACCURACY BUT WE ACHIEVED ALMOST ACCURATE WHILE DETECTING THE FRUITS. WE HAVE DONE SOME MODEL TWERQUING IN THE CODE AND WE HAVE INCREASED THE ACCURACY BY ADAPTING THE DARKNET IN TO THE MODEL

DARKNET: DARKNET IS AN OPEN SOURCE NEURAL NETWORK FRAMEWORK WRITTEN IN C AND CUDA. IT IS FAST, EASY TO INSTALL, AND SUPPORTS CPU AND GPU COMPUTATION.

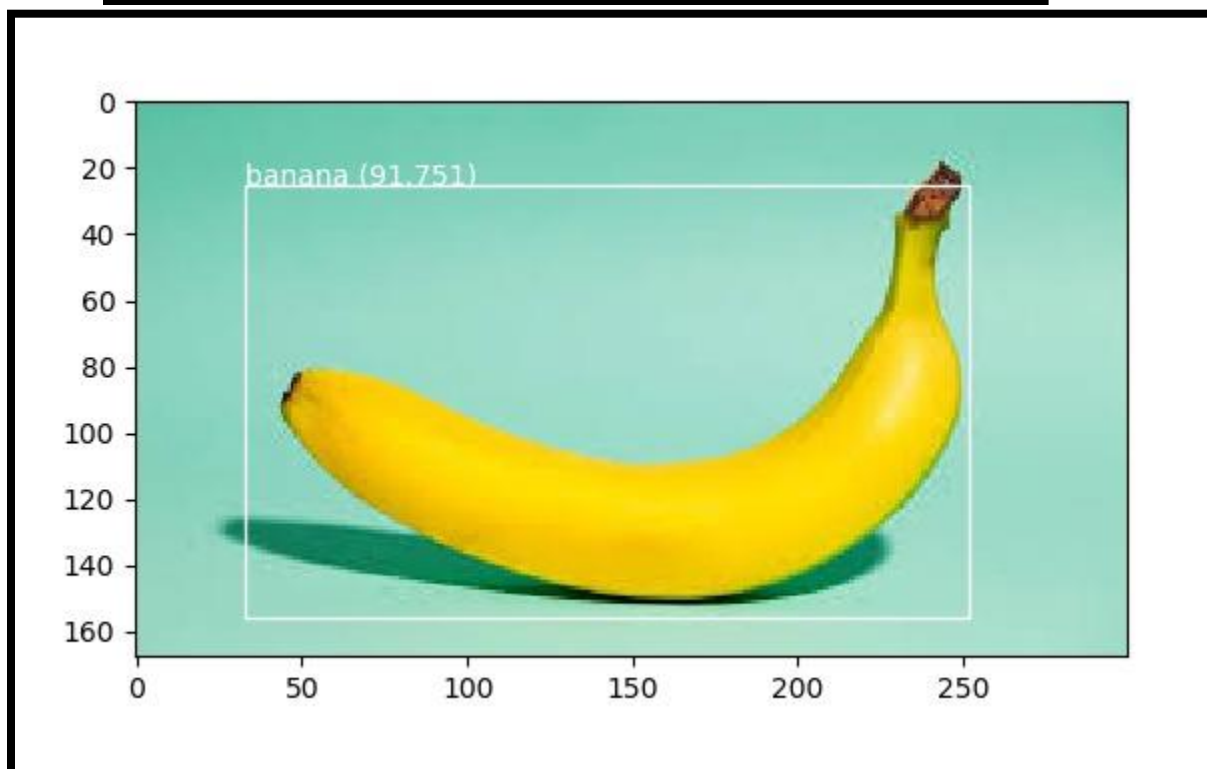
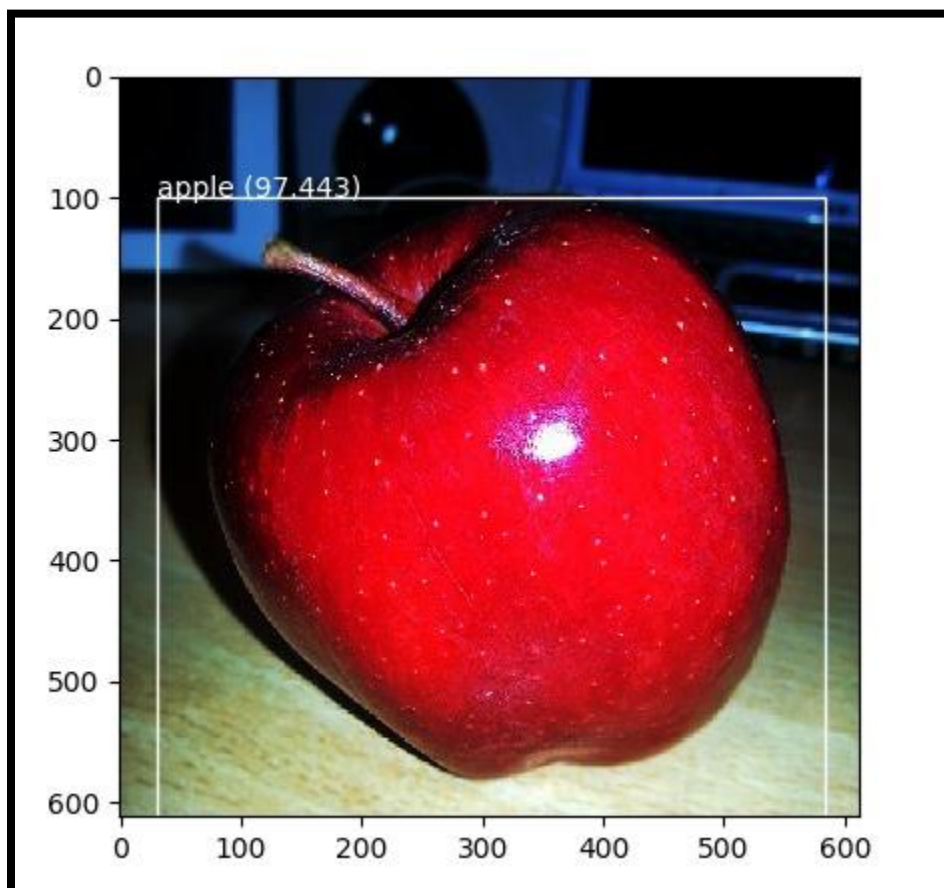
BY ADAPTING TO THIS DARKNET WE HAVE ACHIEVED AN INCREASE IN THE ACCURACY AND ALSO, WE ARE

```
# load yolov3 model
model = load_model('model.h5')
# define the expected input shape for the model
input_w, input_h = 416, 416
# define our new photo
photo_filename = 'apple.jpg'
# load and prepare image
image, image_w, image_h = load_image_pixels(photo_filename, (input_w, input_h))
# make prediction
yhat = model.predict(image)
# summarize the shape of the list of arrays
print([a.shape for a in yhat])
# define the anchors
anchors = [[116,90, 156,198, 373,326], [30,61, 62,45, 59,119], [10,13, 16,30, 33,23]]
# define the probability threshold for detected objects
class_threshold = 0.6
boxes = list()
for i in range(len(yhat)):
    # decode the output of the network
    boxes += decode_netout(yhat[i][0], anchors[i], class_threshold, input_h, input_w)
# correct the sizes of the bounding boxes for the shape of the image
correct_yolo_boxes(boxes, image_h, image_w, input_h, input_w)
# suppress non-maximal boxes
do_nms(boxes, 0.5)
# define the labels
labels = ["person", "bicycle", "car", "motorbike", "aeroplane", "bus", "train", "truck",
"boat", "traffic light", "fire hydrant", "stop sign", "parking meter", "bench",
"bird", "cat", "dog", "horse", "sheep", "cow", "elephant", "bear", "zebra", "giraffe",
"backpack", "umbrella", "handbag", "tie", "suitcase", "frisbee", "skis", "snowboard",
"sports ball", "kite", "baseball bat", "baseball glove", "skateboard", "surfboard",
"tennis racket", "bottle", "wine glass", "cup", "fork", "knife", "spoon", "bowl", "banana",
"apple", "sandwich", "orange", "broccoli", "carrot", "hot dog", "pizza", "donut", "cake",
"chair", "sofa", "pottedplant", "bed", "diningtable", "toilet", "tvmonitor", "laptop", "mouse",
"remote", "keyboard", "cell phone", "microwave", "oven", "toaster", "sink", "refrigerator",
"book", "clock", "vase", "scissors", "teddy bear", "hair drier", "toothbrush"]
# get the details of the detected objects
v_boxes, v_labels, v_scores = get_boxes(boxes, labels, class_threshold)
```

```

171 # load yolov3 model
172 model = load_model('model.h5')
173 # define the expected input shape for the model
174 input_w, input_h = 416, 416
175 # define our new photo
176 photo_filename = 'banana.jpg'
177 # load and prepare image
178 image, image_w, image_h = load_image_pixels(photo_filename, (input_w, input_h))
179 # make prediction
180 yhat = model.predict(image)
181 # summarize the shape of the list of arrays
182 print([a.shape for a in yhat])
183 # define the anchors
184 anchors = [[116,90, 156,198, 373,326], [30,61, 62,45, 59,119], [10,13, 16,30, 33,23]]
185 # define the probability threshold for detected objects
186 class_threshold = 0.6
187 boxes = list()
188 for i in range(len(yhat)):
189     # decode the output of the network
190     boxes += decode_netout(yhat[i][0], anchors[i], class_threshold, input_h, input_w)
191 # correct the sizes of the bounding boxes for the shape of the image
192 correct_yolo_boxes(boxes, image_h, image_w, input_h, input_w)
193 # suppress non-maximal boxes
194 do_nms(boxes, 0.5)
195 # define the labels
196 labels = ["person", "bicycle", "car", "motorbike", "aeroplane", "bus", "train", "truck",
197          "boat", "traffic light", "fire hydrant", "stop sign", "parking meter", "bench",
198          "bird", "cat", "dog", "horse", "sheep", "cow", "elephant", "bear", "zebra", "giraffe",
199          "backpack", "umbrella", "handbag", "tie", "suitcase", "frisbee", "skis", "snowboard",
200          "sports ball", "kite", "baseball bat", "baseball glove", "skateboard", "surfboard",
201          "tennis racket", "bottle", "wine glass", "cup", "fork", "knife", "spoon", "bowl", "banana",
202          "apple", "sandwich", "orange", "broccoli", "carrot", "hot dog", "pizza", "donut", "cake",
203          "chair", "sofa", "pottedplant", "bed", "diningtable", "toilet", "tvmonitor", "laptop", "mouse",
204          "remote", "keyboard", "cell phone", "microwave", "oven", "toaster", "sink", "refrigerator",
205          "book", "clock", "vase", "scissors", "teddy bear", "hair drier", "toothbrush"]
206 # get the details of the detected objects
207 v_boxes, v_labels, v_scores = get_boxes(boxes, labels, class_threshold)

```

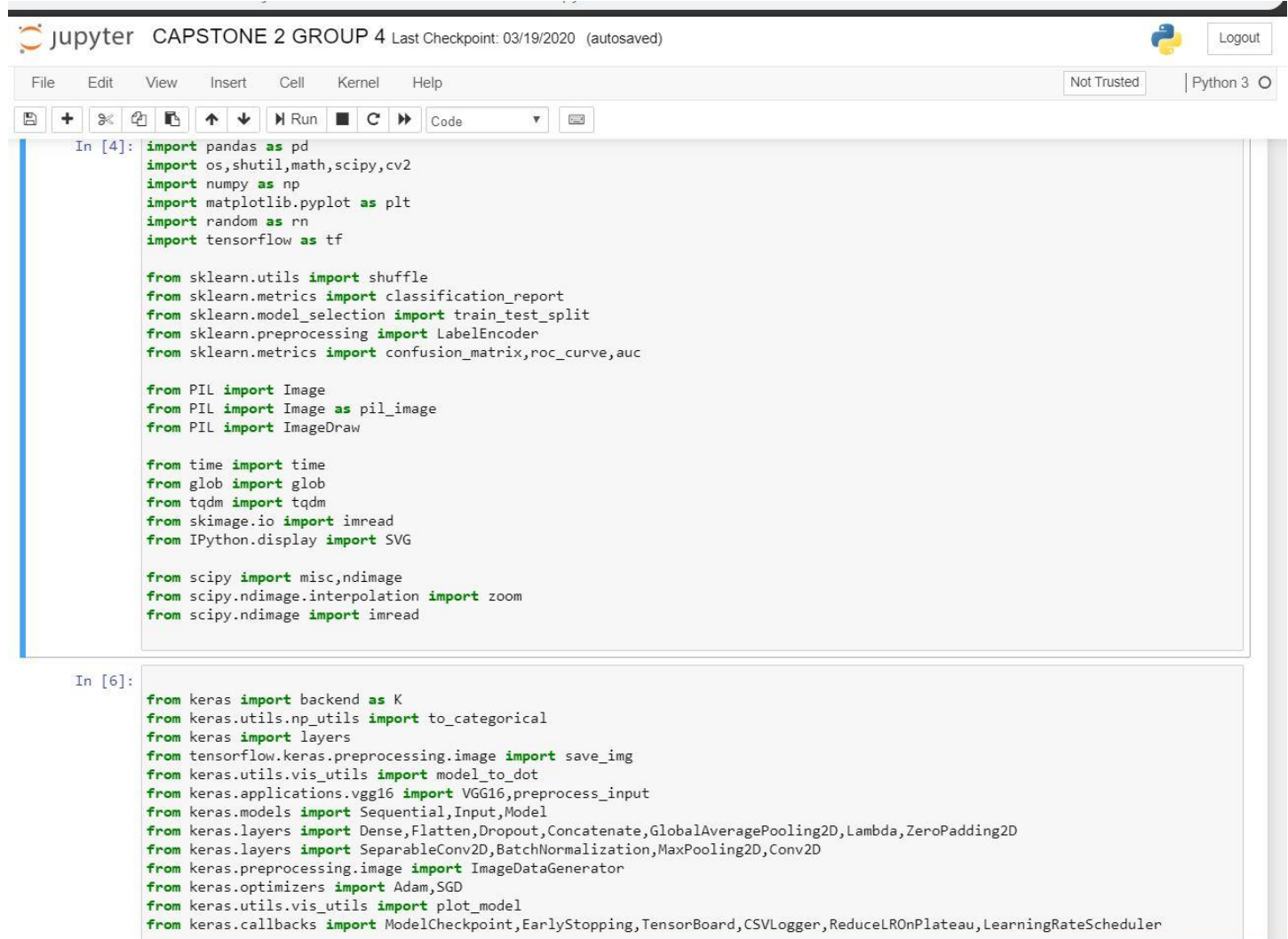



ALTERNATIVES:

WE ARE THINKING OF POSSIBLE ALTERNATIVE WHICH IS "IMAGENET" IN FUTURE IF WE DID NOT ACHIEVED 100 PERCENT ACCURACY USING THE DARKNET. THE IMAGENET PROJECT IS A LARGE VISUAL DATABASE DESIGNED FOR USE IN VISUAL OBJECT RECOGNITION SOFTWARE RESEARCH. MORE THAN 14 MILLION IMAGES HAVE BEEN HAND-ANNOTATED BY THE PROJECT TO INDICATE WHAT OBJECTS ARE PICTURED AND IN AT LEAST ONE MILLION OF THE IMAGES, BOUNDING BOXES ARE ALSO PROVIDED. IMAGENET CONTAINS MORE THAN 20,000 CATEGORIES WITH A TYPICAL CATEGORY, SUCH AS "BALLOON" OR "STRAWBERRY", CONSISTING OF SEVERAL HUNDRED IMAGES. THE DATABASE OF ANNOTATIONS OF THIRD-PARTY IMAGE URLS IS FREELY AVAILABLE DIRECTLY FROM IMAGENET, THOUGH THE ACTUAL IMAGES ARE NOT OWNED BY IMAGENET. SINCE 2010, THE IMAGENET PROJECT RUNS AN ANNUAL SOFTWARE CONTEST, THE IMAGENET LARGE SCALE VISUAL RECOGNITION CHALLENGE (ILSVRC), WHERE SOFTWARE PROGRAMS COMPETE TO CORRECTLY CLASSIFY AND DETECT OBJECTS AND SCENES. THE CHALLENGE USES A "TRIMMED" LIST OF ONE THOUSAND NON-OVERLAPPING CLASSES.

PROJECT OUTCOMES:**ALGORITHM:**

WE STARTED WITH LOADING ALL THE LIBRARIES AND DEPENDENCIES.

A screenshot of a Jupyter Notebook interface. The top bar shows the Jupyter logo, the text "CAPSTONE 2 GROUP 4", and "Last Checkpoint: 03/19/2020 (autosaved)". On the right, there is a Python logo and a "Logout" button. Below the top bar is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", and "Help". To the right of the menu bar are "Not Trusted" and "Python 3" indicators. Below the menu bar is a toolbar with icons for saving, opening, and running code. The main area contains two code cells. The first cell, labeled "In [4]:", contains a list of import statements for various libraries including pandas, os, shutil, math, scipy, cv2, numpy, matplotlib, random, tensorflow, sklearn, PIL, time, glob, tqdm, skimage, IPython, and scipy. The second cell, labeled "In [6]:", contains a list of import statements for Keras and its various components, including backend, utils, layers, models, and callbacks.

```
In [4]: import pandas as pd
import os,shutil,math,scipy,cv2
import numpy as np
import matplotlib.pyplot as plt
import random as rn
import tensorflow as tf

from sklearn.utils import shuffle
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix,roc_curve, auc

from PIL import Image
from PIL import Image as pil_image
from PIL import ImageDraw

from time import time
from glob import glob
from tqdm import tqdm
from skimage.io import imread
from IPython.display import SVG

from scipy import misc,ndimage
from scipy.ndimage.interpolation import zoom
from scipy.ndimage import imread

In [6]: from keras import backend as K
from keras.utils.np_utils import to_categorical
from keras import layers
from tensorflow.keras.preprocessing.image import save_img
from keras.utils.vis_utils import model_to_dot
from keras.applications.vgg16 import VGG16,preprocess_input
from keras.models import Sequential,Input,Model
from keras.layers import Dense,Flatten,Dropout,Concatenate,GlobalAveragePooling2D,Lambda,ZeroPadding2D
from keras.layers import SeparableConv2D,BatchNormalization,MaxPooling2D,Conv2D
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import Adam,SGD
from keras.utils.vis_utils import plot_model
from keras.callbacks import ModelCheckpoint,EarlyStopping,TensorBoard,CSVLogger,ReduceLROnPlateau, LearningRateScheduler
```

NEXT WE MADE A FUNCTION TO PLOT LOSS AND ACCURACY WHICH WE WILL GET AFTER TRAINING THE MODEL AND MADE A COUPLE OF FUNCTIONS TO RETURN THE LABEL OF A PARTICULAR IMAGE AND LOAD THE DATASET THEN WE CONTINUED WITH MAKING SEPARATE CLASSES FOR EACH AND EVERY PRODUCT AND CALLED THE FUNCTION TO LOAD ALL THE IMAGES FOR EVERY CLASS.

```
In [7]: def show_final_history(history):
fig, ax = plt.subplots(1, 2, figsize=(15,5))
ax[0].set_title('loss')
ax[0].plot(history.epoch, history.history["loss"], label="Train loss")
ax[0].plot(history.epoch, history.history["val_loss"], label="Validation loss")
ax[1].set_title('acc')
ax[1].plot(history.epoch, history.history["acc"], label="Train acc")
ax[1].plot(history.epoch, history.history["val_acc"], label="Validation acc")
ax[0].legend()
ax[1].legend()
```

```
In [8]: def label_assignment(img,label):
return label

def training_data(label,data_dir):
for img in tqdm(os.listdir(data_dir)):
label = label_assignment(img,label)
path = os.path.join(data_dir,img)
img = cv2.imread(path,cv2.IMREAD_COLOR)
img = cv2.resize(img,(imgsize,imgsize))

X.append(np.array(img))
Z.append(str(label))
```

NEXT WE CONVERTED THE IMAGES TO CATEGORICAL FORMAT BEFORE FINALLY NORMALIZING THE IMAGES AND SPLIT THE DATASET INTO 2 PARTS TRAINING SET WITH 80% OF DATA AND TESTING SET WITH 20%.SINCE THE DATASET SIZE WAS QUITE SMALL, WE USED DATA AUGMENTATION STRATEGIES LIKE ROTATION, ZOOMING AND FLIPPING. THIS ALMOST DOUBLED THE SIZE OF DATASET.

```
In [14]: label_encoder= LabelEncoder()
Y = label_encoder.fit_transform(Z)
Y = to_categorical(Y,25)
X = np.array(X)
X=X/255

x_train,x_test,y_train,y_test = train_test_split(X,Y,test_size=0.2,random_state=42)
```

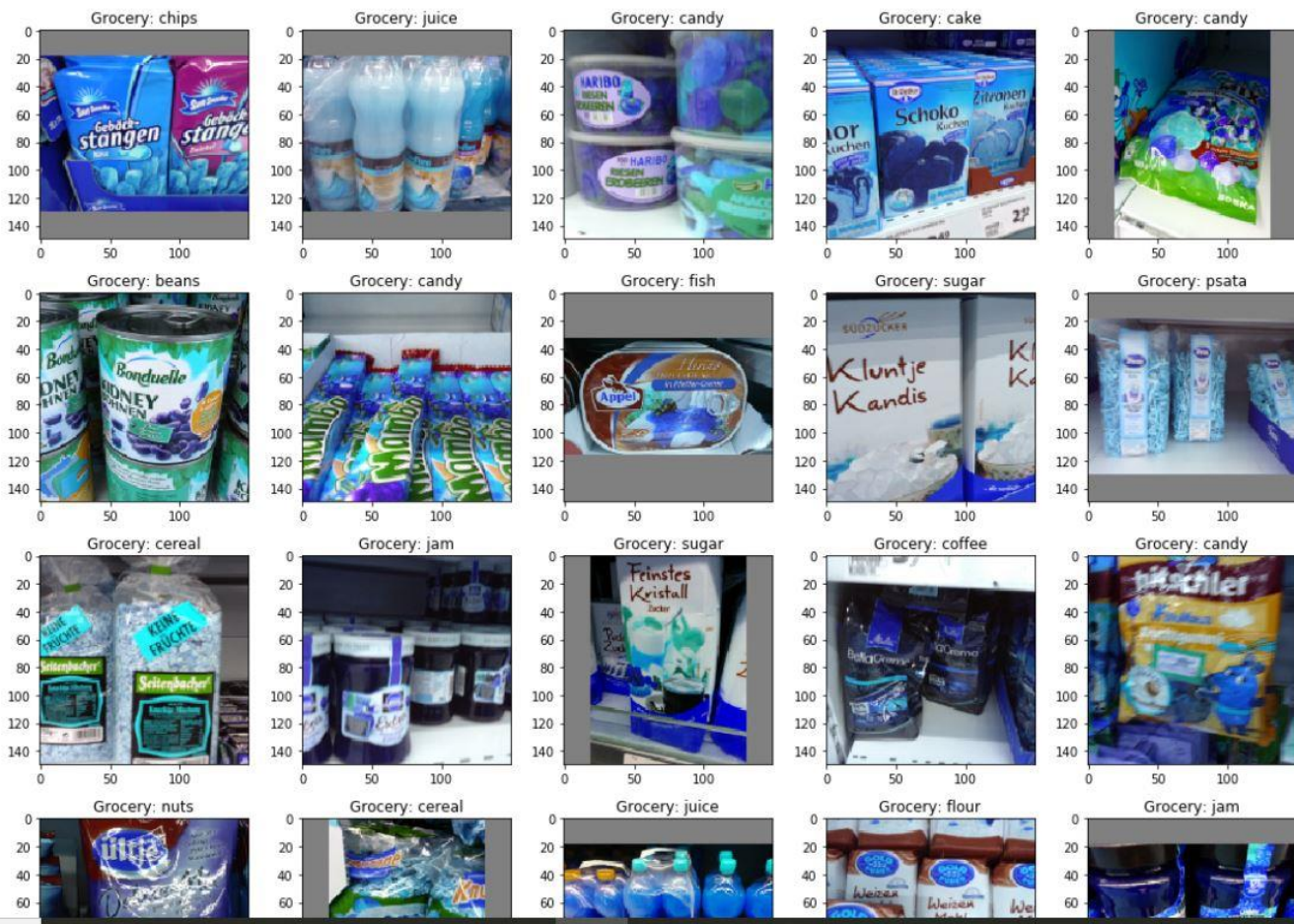
```
In [15]: augs_gen = ImageDataGenerator(
featurewise_center=False,
samplewise_center=False,
featurewise_std_normalization=False,
samplewise_std_normalization=False,
zca_whitening=False,
rotation_range=10,
zoom_range = 0.1,
width_shift_range=0.2,
height_shift_range=0.2,
horizontal_flip=True,
vertical_flip=False)

augs_gen.fit(x_train)
```

```
In [16]: fig,ax=plt.subplots(5,5)
fig.set_size_inches(15,15)
for i in range(5):
for j in range(5):
l=mn.randint(0,len(Z))
ax[i,j].imshow(X[l])
ax[i,j].set_title('Grocery: '+Z[l])

plt.tight_layout()
```

SOME SAMPLE EXAMPLES FROM EACH CATEGORY



VISUALIZING THE MODEL

```
In [17]: base_model = VGG16(include_top=False,
                             input_shape = (imgsize,imgsize,3),
                             weights = 'imagenet')

for layer in base_model.layers:
    layer.trainable = False

for layer in base_model.layers:
    print(layer,layer.trainable)

model = Sequential()
model.add(base_model)
model.add(GlobalAveragePooling2D())
model.add(Dropout(0.3))
model.add(Dense(25,activation='softmax'))
model.summary()

SVG(model_to_dot(model).create(prog='dot', format='svg'))
plot_model(model, to_file='model_plot.png', show_shapes=True, show_layer_names=True)
```

WARNING:tensorflow:From C:\Users\User\Anaconda3\envs\Delia\lib\site-packages\keras\backend\tensorflow_backend.py:58: The name tf.get_default_graph is deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From C:\Users\User\Anaconda3\envs\Delia\lib\site-packages\keras\backend\tensorflow_backend.py:341: The name tf.variables_initializer is deprecated. Please use tf.compat.v1.variables_initializer instead.

```
<keras.engine.topology.InputLayer object at 0x000001E763024248> False
<keras.layers.convolutional.Conv2D object at 0x000001E74E5E9F08> False
<keras.layers.convolutional.Conv2D object at 0x000001E74E5E9808> False
<keras.layers.pooling.MaxPooling2D object at 0x000001E76357DC08> False
<keras.layers.convolutional.Conv2D object at 0x000001E762815A48> False
<keras.layers.convolutional.Conv2D object at 0x000001E762816688> False
<keras.layers.pooling.MaxPooling2D object at 0x000001E76281DBC8> False
<keras.layers.convolutional.Conv2D object at 0x000001E762822B88> False
<keras.layers.convolutional.Conv2D object at 0x000001E762820648> False
<keras.layers.convolutional.Conv2D object at 0x000001E762828B08> False
<keras.layers.pooling.MaxPooling2D object at 0x000001E76282CEC8> False
<keras.layers.convolutional.Conv2D object at 0x000001E762832E48> False
<keras.layers.convolutional.Conv2D object at 0x000001E762834988> False
<keras.layers.convolutional.Conv2D object at 0x000001E762838F48> False
<keras.layers.pooling.MaxPooling2D object at 0x000001E76283E288> False
<keras.layers.convolutional.Conv2D object at 0x000001E762841DC8> False
<keras.layers.convolutional.Conv2D object at 0x000001E762842D08> False
<keras.layers.convolutional.Conv2D object at 0x000001E76284C288> False
<keras.layers.pooling.MaxPooling2D object at 0x000001E762854608> False
WARNING:tensorflow:From C:\Users\User\Anaconda3\envs\Delia\lib\site-packages\keras\backend\tensorflow_backend.py:2888: calling dropout (from tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
```

Layer (type)	Output Shape	Param #
vgg16 (Model)	(None, 4, 4, 512)	14714688
global_average_pooling2d_1 ((None, 512)		0
dropout_1 (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 25)	12825
Total params: 14,727,513		
Trainable params: 12,825		
Non-trainable params: 14,714,688		

WE DEFINED THE 2 OPTIMIZERS. FIRST ONE BEING A STOCHASTIC GRADIENT DESCENT WITH A LEARNING RATE OF 0.0001 AND MOMENTUM VALUE OF 0.99. THE SECOND ONE WAS ADAM WITH A SLIGHTLY LARGER LEARNING RATE OF 0.001. ON COMPARING THE TWO, WE FOUND THAT ADAM WAS GIVING MUCH BETTER RESULTS. ALSO, I HAVE USED CATEGORICAL CROSS-ENTROPY AS THE LOSS FUNCTION FOR OUR MODEL. I TRAINED THE MODEL FOR 50 ITERATIONS WITH A BATCH SIZE OF 128 WHICH I FIGURED OUT TO BE THE OPTIMAL ONE FOR THIS PROBLEM.

```
In [19]: opt = SGD(lr=1e-4,momentum=0.99)
opt1 = Adam(lr=1e-3)

model.compile(
    loss='categorical_crossentropy',
    optimizer=opt1,
    metrics=['accuracy']
)

history = model.fit_generator(
    aug_gen.flow(x_train,y_train,batch_size=128),
    validation_data = (x_test,y_test),
    validation_steps = 100,
    steps_per_epoch = 100,
    epochs = 50,
    verbose = 1,
    callbacks=callbacks
)

Epoch 45/50
99/100 [=====>.] - ETA: 0s - loss: 1.5287 - acc: 0.5422Epoch 00044: val_acc did not improve
100/100 [=====>.] - 33s - loss: 1.5289 - acc: 0.5423 - val_loss: 1.4667 - val_acc: 0.5626
Epoch 46/50
99/100 [=====>.] - ETA: 0s - loss: 1.5220 - acc: 0.5480Epoch 00045: val_acc improved from 0.56869 to
0.57172, saving model to ./base.model
100/100 [=====>.] - 33s - loss: 1.5214 - acc: 0.5481 - val_loss: 1.4623 - val_acc: 0.5717
Epoch 47/50
99/100 [=====>.] - ETA: 0s - loss: 1.5111 - acc: 0.5482Epoch 00046: val_acc did not improve
100/100 [=====>.] - 33s - loss: 1.5122 - acc: 0.5479 - val_loss: 1.4541 - val_acc: 0.5667
Epoch 48/50
99/100 [=====>.] - ETA: 0s - loss: 1.5156 - acc: 0.5442Epoch 00047: val_acc improved from 0.57172 to
0.57374, saving model to ./base.model
100/100 [=====>.] - 33s - loss: 1.5143 - acc: 0.5444 - val_loss: 1.4542 - val_acc: 0.5737
Epoch 49/50
99/100 [=====>.] - ETA: 0s - loss: 1.4989 - acc: 0.5510Epoch 00048: val_acc did not improve
100/100 [=====>.] - 33s - loss: 1.4992 - acc: 0.5512 - val_loss: 1.4527 - val_acc: 0.5727
Epoch 50/50
99/100 [=====>.] - ETA: 0s - loss: 1.5053 - acc: 0.5420Epoch 00049: val_acc did not improve
100/100 [=====>.] - 33s - loss: 1.5063 - acc: 0.5418 - val_loss: 1.4493 - val_acc: 0.5697
```

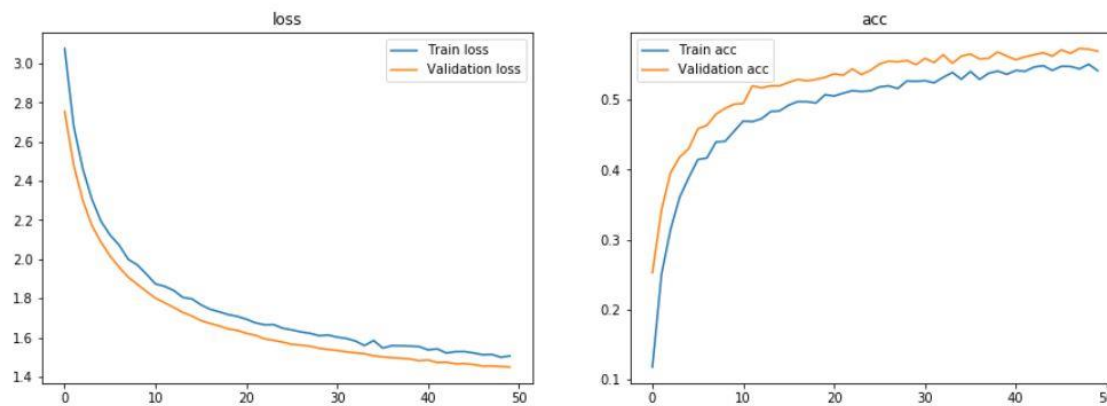
FINALLY, WE PLOTTED THE RESULTS

```
In [20]: show_final_history(history)
model.load_weights('./base.model')

model_json = model.to_json()
with open("model.json","w") as json_file:
    json_file.write(model_json)

model.save("model.h5")
print("Weights Saved")
```

Weights Saved



In []:

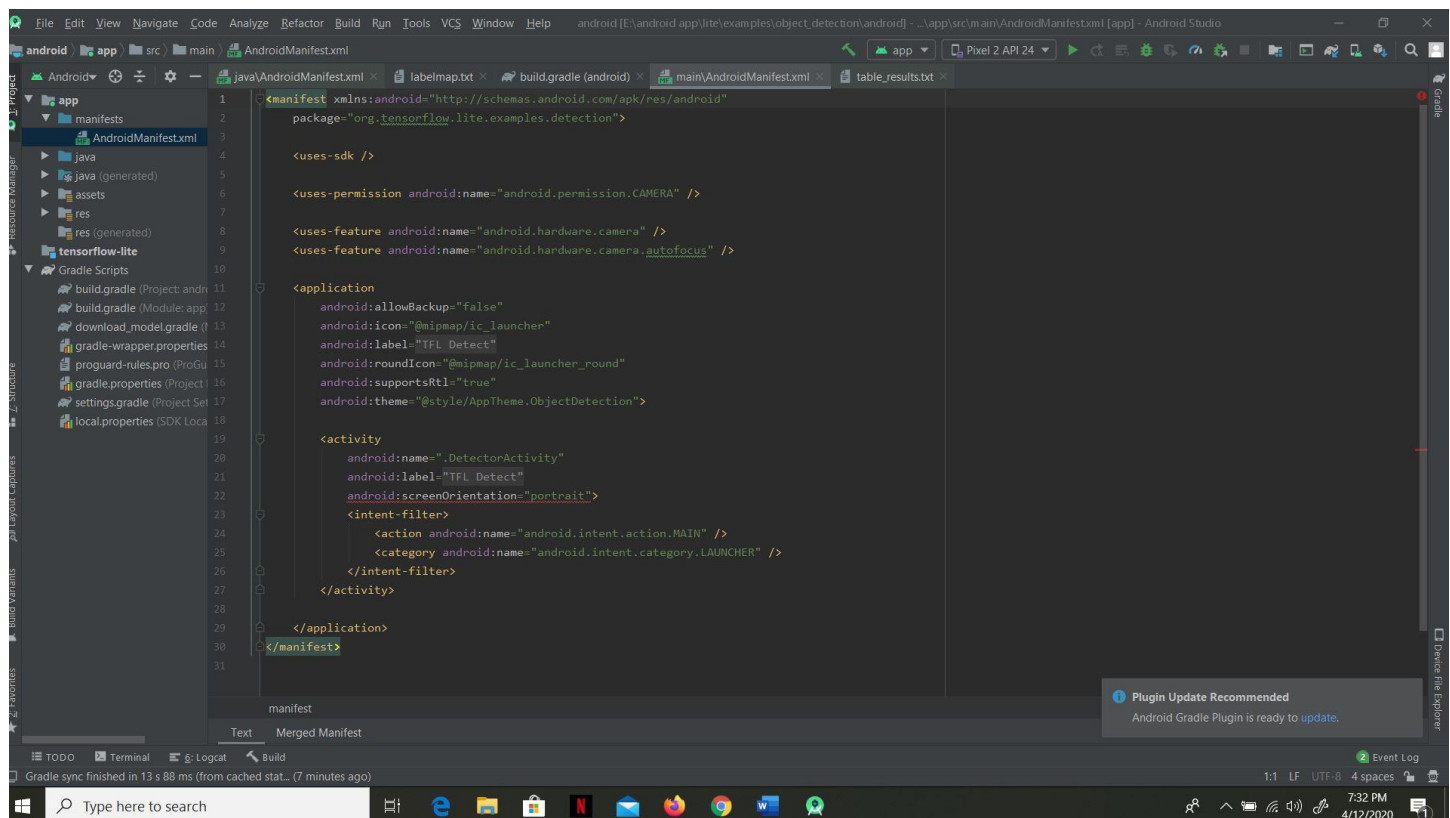
THE MODEL IS ABLE TO REACH A VALIDATION ACCURACY OF 60% WHICH IS QUITE GOOD CONSIDERING THE NUMBER OF CLASSES (25) WITH 100–200 IMAGES IN EACH CATEGORY.

MOBILE APP:

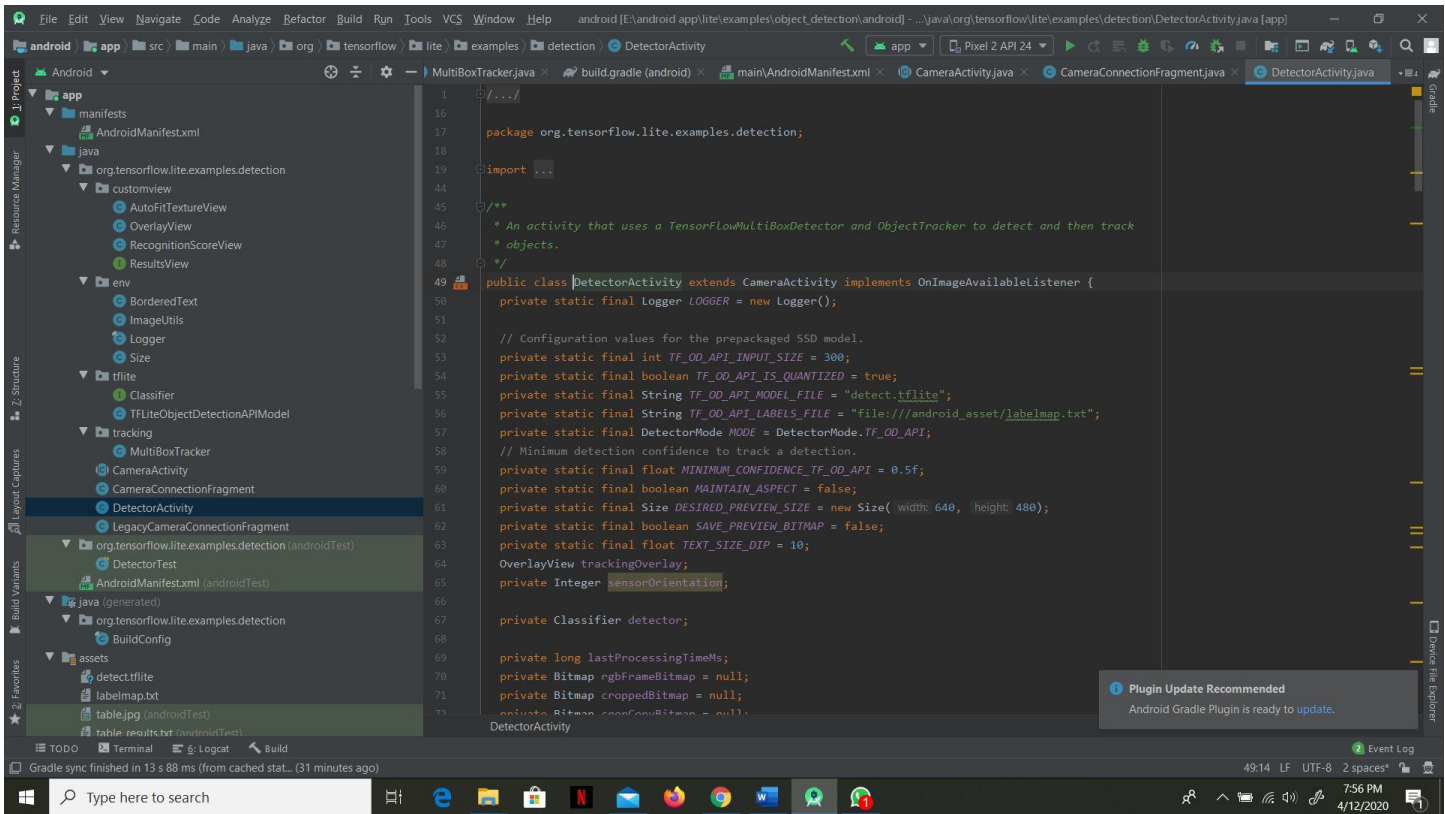
FOR THE MOBILE APP WE USED ANDROID AS OUR PLATFORM WHICH GAVE US EASY WAY TO RATE THE APP AND ALSO INTEGRATION IS SUCCESSFUL INDIVIDUALLY FOR DATABASE AND ALGORITHM. WE HAVE USED JAVA, PHP, PYTHON, C++ TO CREATE THE APP.

THE BELOW IMAGE IS THE ANDROIDMANIFEST.XML FILE WHICH CONTAINS INFORMATION OF THE APP PACKAGES, INCLUDING COMPONENTS OF THE APPLICATION SUCH AS ACTIVITIES, SERVICES, BROADCAST RECEIVERS, CONTENT PROVIDERS ETC. IT IS RESPONSIBLE TO PROTECT THE APPLICATION TO ACCESS ANY PROTECTED PARTS BY PROVIDING THE PERMISSIONS.

AND ALSO DECLARES THE ANDROID API THAT THE APPLICATION IS GOING TO USE. THIS MANIFEST FILE LISTS THE INSTRUMENTATION CLASSES. THE INSTRUMENTATION CLASSES PROVIDES PROFILING AND OTHER INFORMATIONS. THESE INFORMATIONS ARE REMOVED JUST BEFORE THE APPLICATION IS PUBLISHED ETC.



THIS IS HOW WE HVE USED TENSORFLOW LITE FOR THE DETECTION OF THE OBJECTS USING THE ANDROID APP AND THE INBUILT CAMERA



```

1  //...
16
17 package org.tensorflow.lite.examples.detection;
18
19 import
44
45 /**
46  * An activity that uses a TensorFlowMultiBoxDetector and ObjectTracker to detect and then track
47  * objects.
48  */
49 public class DetectorActivity extends CameraActivity implements OnImageAvailableListener {
50     private static final Logger LOGGER = new Logger();
51
52     // Configuration values for the prepackaged SSD model.
53     private static final int TF_OD_API_INPUT_SIZE = 300;
54     private static final boolean TF_OD_API_IS_QUANTIZED = true;
55     private static final String TF_OD_API_MODEL_FILE = "detect.tflite";
56     private static final String TF_OD_API_LABELS_FILE = "file:///android_asset/labelmap.txt";
57     private static final DetectorMode MODE = DetectorMode.TF_OD_API;
58     // Minimum detection confidence to track a detection.
59     private static final float MINIMUM_CONFIDENCE_TF_OD_API = 0.5f;
60     private static final boolean MAINTAIN_ASPECT = false;
61     private static final Size DESIRED_PREVIEW_SIZE = new Size( width: 640, height: 480);
62     private static final boolean SAVE_PREVIEW_BITMAP = false;
63     private static final float TEXT_SIZE_DIP = 10;
64     OverlayView trackingOverlay;
65     private Integer sensorOrientation;
66
67     private Classifier detector;
68
69     private long lastProcessingTimeMs;
70     private Bitmap rgbFrameBitmap = null;
71     private Bitmap croppedBitmap = null;
72     private Bitmap cropFrameBitmap = null;
73
74     DetectorActivity
    
```

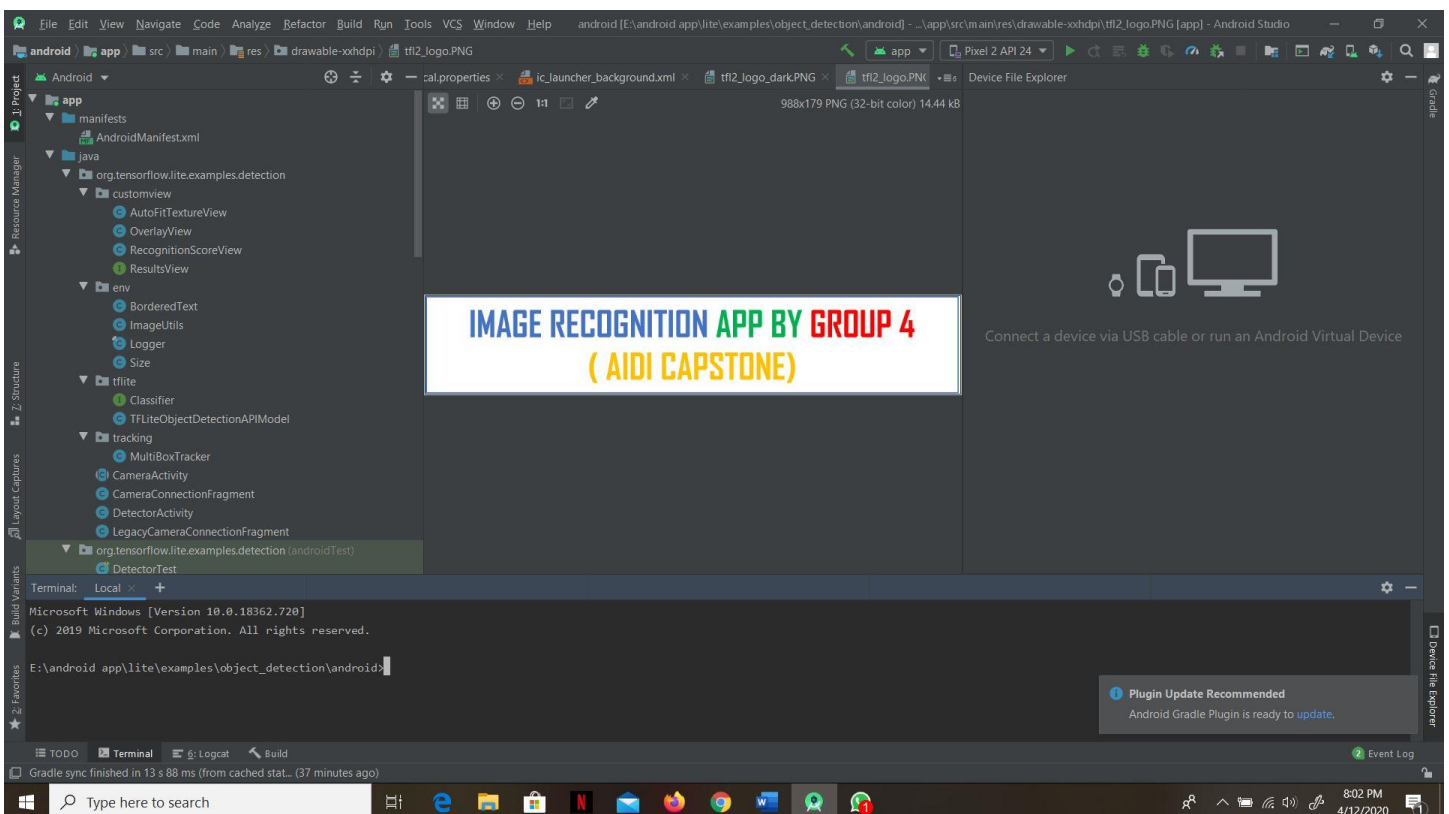
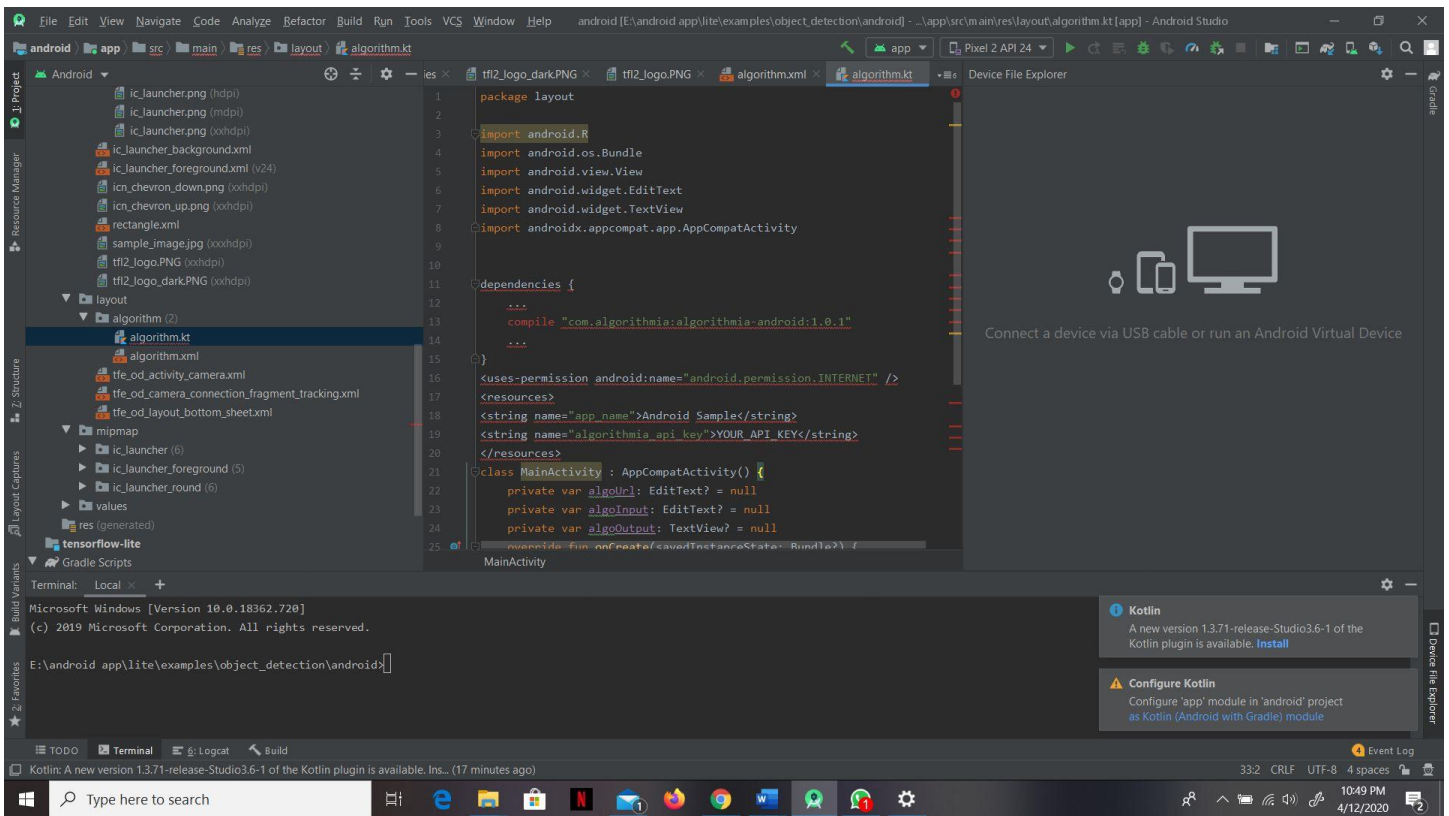
THIS IS WHERE WE HAVE INTEGRATED OUR ALGORITHM IN THE ANDROID APP USING THE ALGORITHM JAVA CLIENT CODE WHICH WE HAVE WRITTEN TO INTEGRATE THE ALGORITHM WITH OUR APP TO DETECT THE OBJECT WITH THE HELP OF THE CAMERA. WE USED TENSORFLOW PLUGINS TO INTEGRATE THE ALGORITHM AND THE APP

WHAT THE ABOVE CODE DOES IS SET THE ALGOOUTPUT TEXT VIEW TO DISPLAY THE RESULTS OF THE CALL. IF THE RESPONSE IS NULL, WE'VE SET THE OUTPUT TO SHOW A GENERIC "NETWORK CONNECTION FAILED" MESSAGE. FOLLOWING THIS, WE'VE GOT A CHECK ON LINE 5 TO SEE IF THE CONDITION IS SUCCESSFUL, AND IF SO, WE'LL SET THE TEXT TO THE SUCCESS RESPONSE. OF COURSE, WE'LL ALSO NEED TO HANDLE THE FAILURES, WHICH YOU'LL SEE IN THE FINAL ELSE OF THE METHOD ON LINES 8-10.

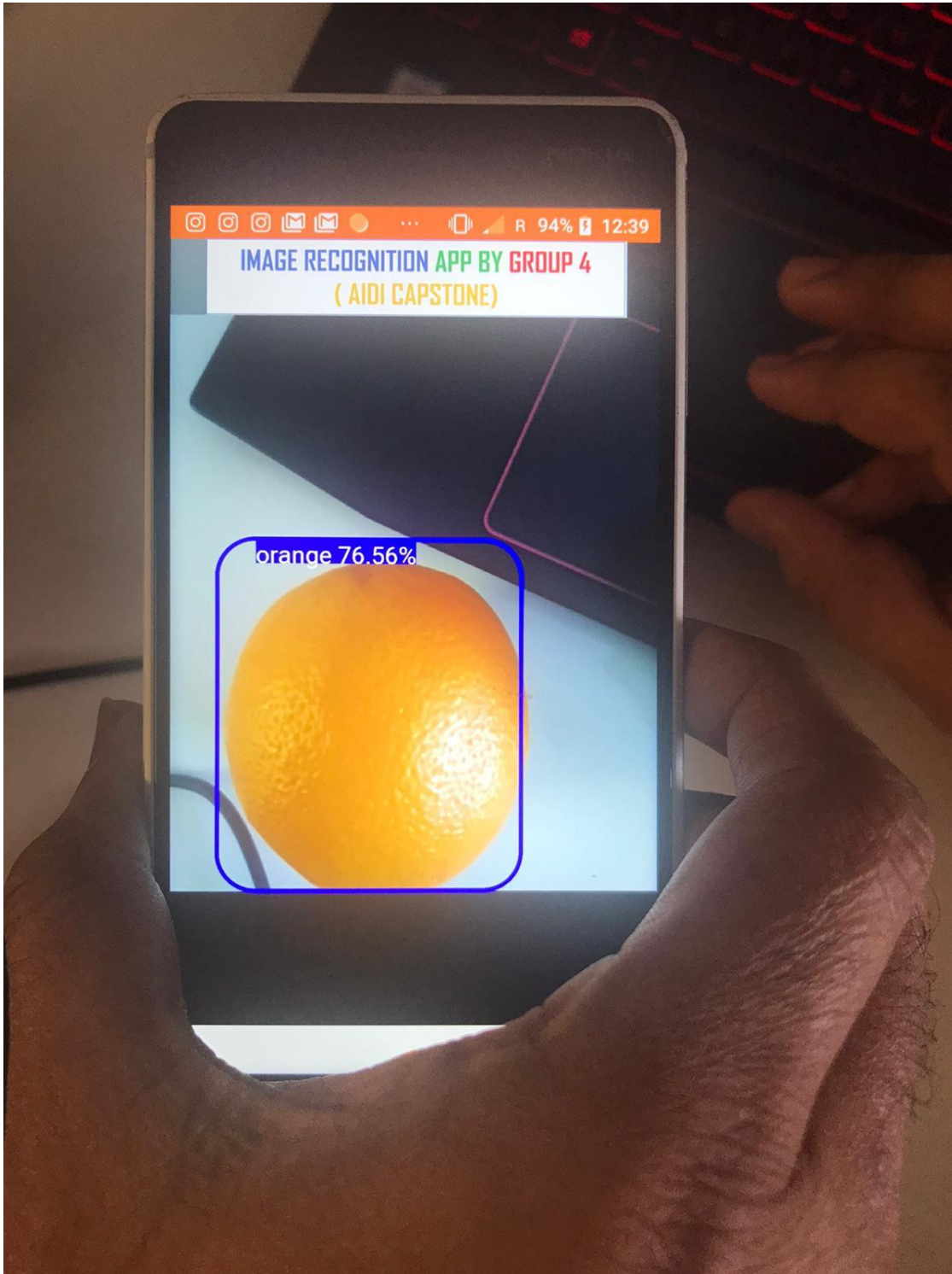
@Override

```

protected void onPostExecute(AlgoResponse response) {
    if(response == null) {
        algoOutput.setText("Algorithm Error: network connection failed");
    } else if(response.isSuccess()) {
        AlgoSuccess success = (AlgoSuccess) response;
        algoOutput.setText(success.asJsonString());
    } else {
        AlgoFailure failure = (AlgoFailure) response;
        algoOutput.setText("Algorithm Error: " + failure.error);
    }
}
    
```

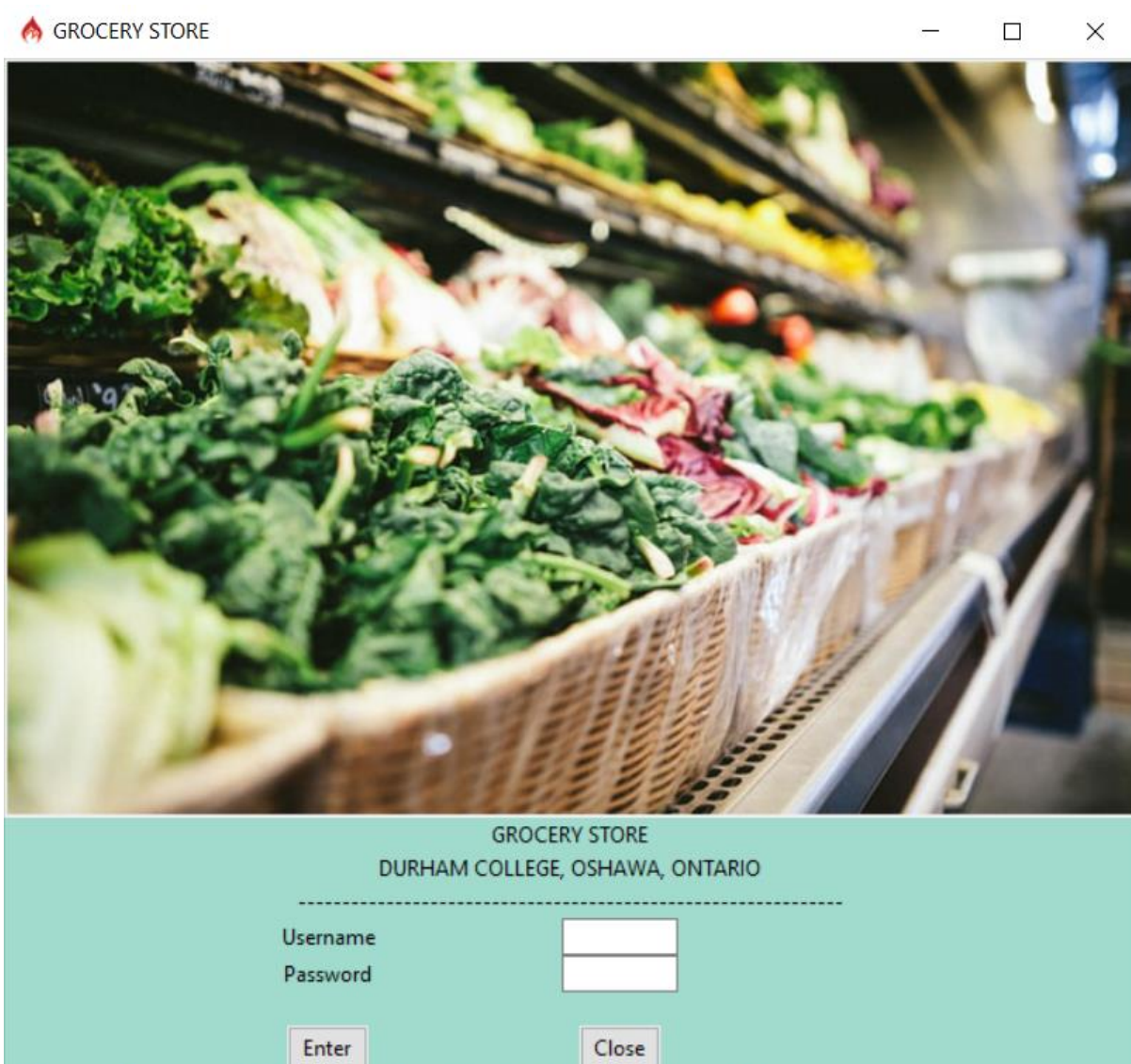



IN THE BELOW IMAGE YOU CAN SEE THE OUTPUT OF OUR PROJECT A WORKING ANDROID APP WHICH CAN DETECT THE OBJECTS WITH MOST OF THEM 70% ACCURACY



INVENTORY MANAGEMENT SYSTEM:

THIS IS THE LOGIN PAGE OF THE APPLICATION. AS OF NOW WE HAVE SET THE USERNAME AND PASSWORD AS DEFAULT TO "ADMIN". IT IS POSSIBLE TO ADD USERS TO THE APPLICATION WITH EACH PERSON HAVING DIFFERENT LOGIN ID'S AND PASSWORD'S.

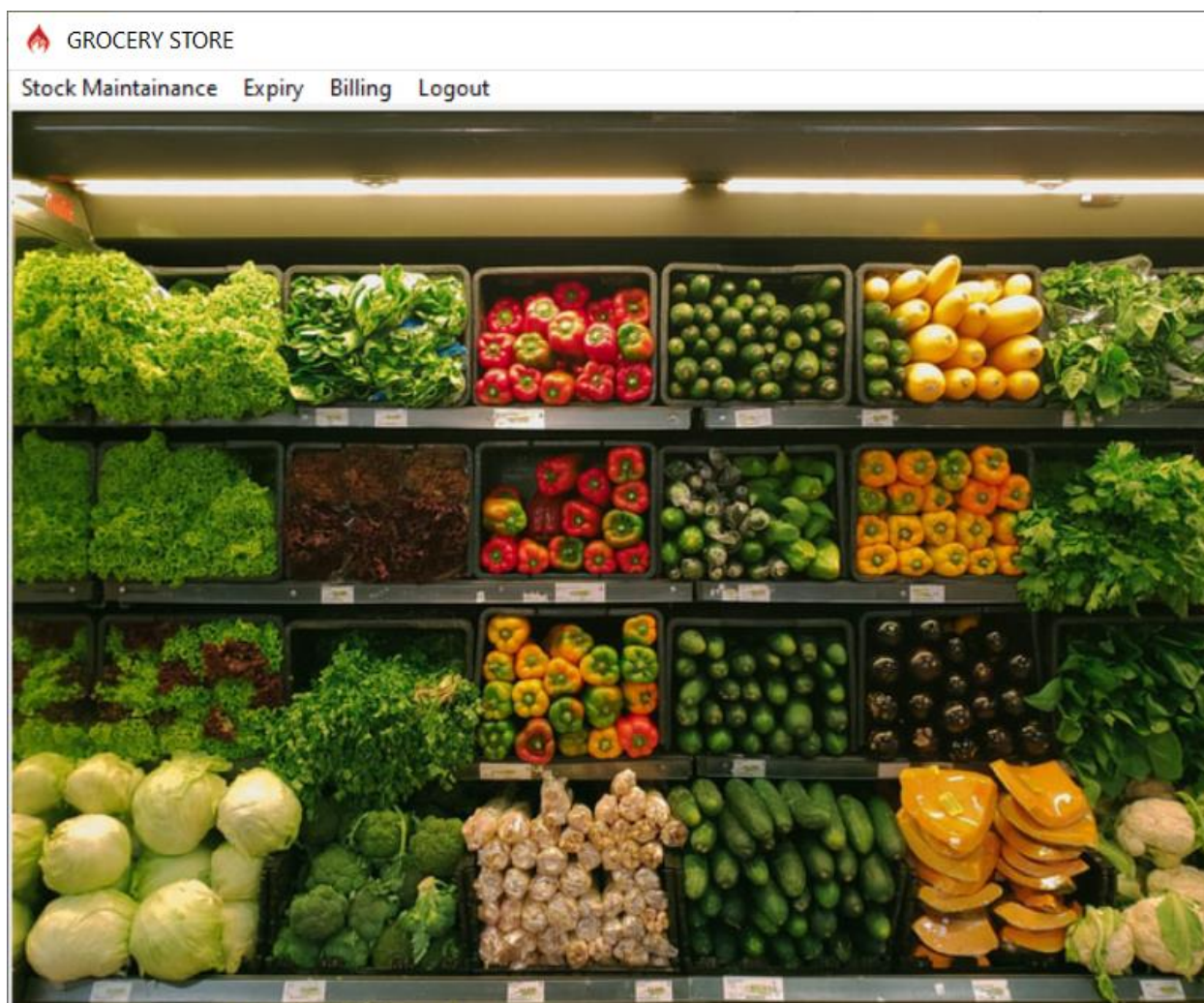


GROCERY STORE

DURHAM COLLEGE, OSHAWA, ONTARIO

Username

Password




AFTER LOGGING IN THE APPLICATION. THE ABOVE IMAGE IS THE HOMEPAGE WHERE WE HAVE DIFFERENT OPTIONS TO PERFORM OPERATIONS LIKE:

- STOCK MAINTAINANCE
- EXPIRY
- BILLING
- LOGOUT

STOCK MAINTAINANCE:


IN THIS WE CAN ADD NEW ITEMS, DELETE ITEMS AND UPDATE DATABASE IN OUR MANAGEMENT SYSTEM.

 Add Stock

Enter a New Item to the Grocery Stock

Item_No:
 Item_Name:
 Item_Type:
 Quantity_Remain:
 Item_Cost:
 Expiry_Date:
 Manufactured_By:

Item_Name	Item_Type	Quantity_Remain	Item_Cost	Expiry_Date	Manufactured_By
1. Milk	Dairy	86	10	12/12/2017	Prairie
2. Sandwich Wheat	Bread	93	9.99	08/31/2017	Essential Everyday
4. Chocalate	MilkShake	341	1.33	12/12/2016	Somya
5. Cake	Milk Product	202	12.99	12/12/2018	Hersheys
6. Marie	Biscuits	122	2.99	12/02/2016	Britannia
7. coke	drink	197	3.99	12/02/2026	coca cola

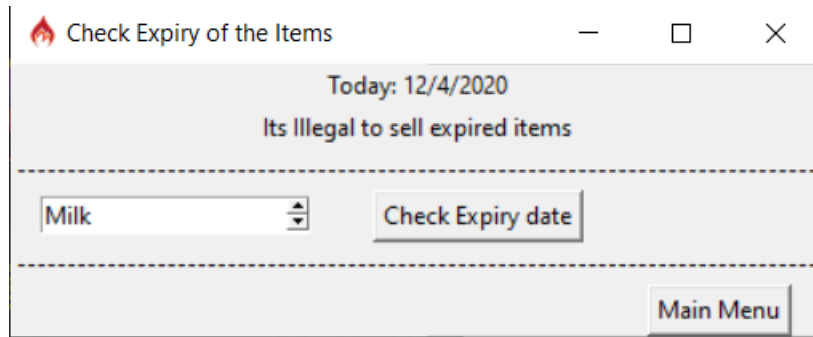
 Delete grocery item from Stock

Enter the Item No to Delete

Item	Qty Remain	Cost	Expiry Date
1) Milk	86	10	12/12/2017
2) Sandwich Wheat	93	9.99	08/31/2017
4) Chocalate	341	1.33	12/12/2016
5) Cake	202	12.99	12/12/2018
6) Marie	122	2.99	12/02/2016
7) coke	197	3.99	12/02/2026

EXPIRY:

THIS FEATURE ADDED TO CHECK THE EXPIRY DATE OF EACH PRODUCT AVAILABLE. AS IT'S ILLEGAL TO SELL EXPIRED PRODUCT.



Check Expiry of the Items

Today: 12/4/2020

Its Illegal to sell expired items

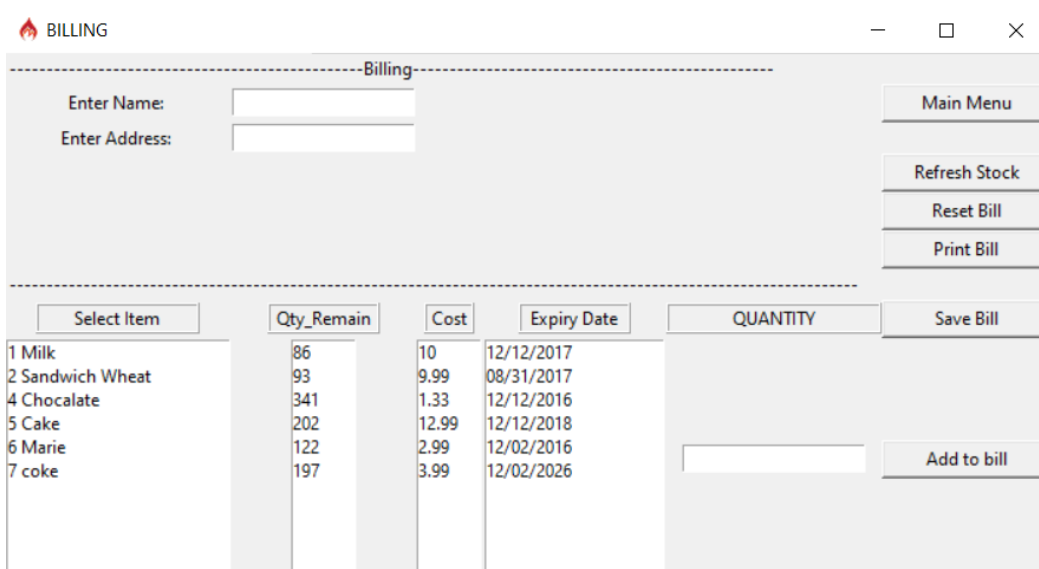
Milk

Check Expiry date

Main Menu

BILLING:

THIS FEATURE GENERATES THE PURCHASE BILL OF THE CUSTOMER. IT IS DESIGNED IN SUCH WAY THAT IT WILL ADD AND CALCULATE THE AMOUNT BY ITSELF. ONLY THING WE MUST GIVE INPUT IS TYPE OF ITEM AND NUMBER OF ITEMS. WE CAN ALSO SEE THE QUANTITY LEFT IN OUR INVENTORY AFTER BILL IS GENERATED. HERE WE HAVE OPTION TO PRINT BILL AS WELL. WE HAVE ALSO CREATED THE OPTION TO CHECK THE DAILY INCOME.



BILLING

Enter Name:

Enter Address:

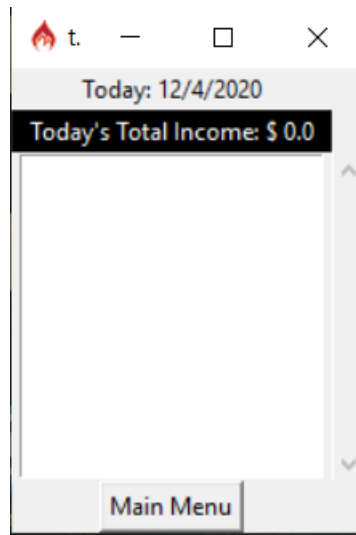
Main Menu

Refresh Stock

Reset Bill

Print Bill

Select Item	Qty_Remain	Cost	Expiry Date	QUANTITY	Save Bill
1 Milk	86	10	12/12/2017		Add to bill
2 Sandwich Wheat	93	9.99	08/31/2017		
4 Chocalate	341	1.33	12/12/2016		
5 Cake	202	12.99	12/12/2018		
6 Marie	122	2.99	12/02/2016		
7 coke	197	3.99	12/02/2026		



BILL GENERATED

```
=====
                                No :985

                GROCERY STORE
            DURHAM COLLEGE, OSHAWA, ONTARIO

=====
Name: Prasad
Address: 380 ARBOR COURT
=====
Product                Qty.      Price
=====
Milk                    3          30.0
Cake                    3          38.97
coke                    3          11.97

=====
Total                    $ 80.94
=====

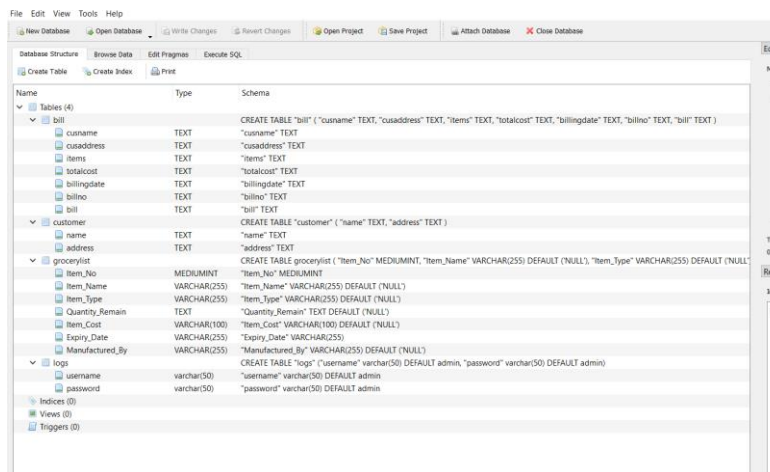
Dealer 's signature:_____
=====
```

LOGOUT:

THIS FEATURE HAS BEEN ADDED SO THAT THE DIFFERENT PEOPLE CAN LOGIN WITH THEIR OWN ID AND PASSWORD.

DATABASE:

WE HAVE USED SQLITE TO CREATE DATABASE FOR THE APPLICATION. WE HAVE CREATED DIFFERENT TABLES SUCH AS:



■ CUSTOMER TABLE: TO STORE CUSTOMERS DATA.

New Database		Open Database		Write Changes		Revert Changes		Open Project		Save Project		Attach Database		Close Database	
Database Structure		Browse Data		Edit Pragmas		Execute SQL									
Table: customer												New Record		Delete R	
name		address													
Filter		Filter													
1	rohit	shardanagar													
2	sana	hyd													
3	rohit	asas													
4	rohit	sharada													
5	santhosh	abc													
6	prasad	380 arbor court													

■ BILLING TABLE: TO STORE BILLING DETAILS.

Database Structure Browse Data Edit Pragmas Execute SQL

Table: bill

	cusname	cusaddress	items	totalcost	billingdate	billno	bill
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	Rohit	yeola road	[1, 2]	69.95	23/3/2020	163	
2	Rohit	Shardanagar	[1]	20.0	23/3/2020	574	
3	Apurva	Springfield, Illi...	[1]	20.0	23/3/2020	361	
4	Apurva	Springfield, Ill...	[1]	40.0	23/3/2020	655	
5	SUmit	srinagar	[1]	40.0	23/3/2020	628	
6	Rohit	Shardanagar	[1]	30.0	23/3/2020	324	
7	rohit	asas	[1, 3, 5]	57.96	23/3/2020	155	
8	Rohit	Sharada	[4, 2]	22.64	23/3/2020	314	
9	santhosh	abc	[1, 3]	31.98	8/4/2020	140	
10	Prasad	380 ARBOR C...	[1, 5, 7]	80.94	8/4/2020	985	

■ STOCK TABLE: TO STORE ALL THE DATA REGARDING STOCK AVAILABLE AND SOLD.

File Edit View Tools Help

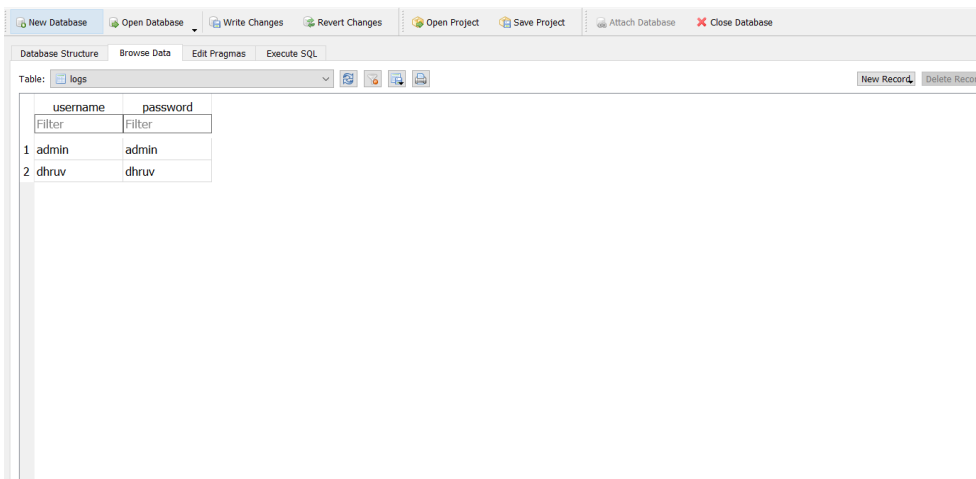
New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragmas Execute SQL

Table: grocerylist

	Item_No	Item_Name	Item_Type	quantity_Remain	Item_Cost	Expiry_Date	Manufactured_B
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Milk	Dairy	86	10	12/12/2017	Prairie
2	2	Sandwich Wh...	Bread	93	9.99	08/31/2017	Essential Eve...
3	4	Chocalate	MilkShake	341	1.33	12/12/2016	Somya
4	5	Cake	Milk Product	202	12.99	12/12/2018	Hersheys
5	6	Marie	Biscuits	122	2.99	12/02/2016	Britannia
6	7	coke	drink	197	3.99	12/02/2026	coca cola

- LOG TABLE: TO STORE ALL THE ID AND PASSWORD OF THE PERSON OR MANAGER.



The screenshot shows a database management interface with a table named 'logs'. The table has two columns: 'username' and 'password'. There are two rows of data: one for 'admin' and one for 'dhruv'.

	username	password
1	admin	admin
2	dhruv	dhruv

PYTHON CODE FOR OUR INVENTORY MANAGEMENT:

LOGIN PAGE:

```

1 #Name - Dhruv Loya
2 #Email - Dhruva.loya@dcmail.ca
3 #Python 3.7
4
5 #Login code
6
7 from tkinter import *
8 from sqlite3 import dbapi2 as sqlite
9 from PIL import ImageTk, Image
10 from tkinter.ttk import *
11
12 login=sqlite.connect("grocery.sqlite")
13 l=login.cursor()
14 WinStat = ''
15
16
17 def stock():
18     application.destroy()
19     login.close()
20
21     import stockdetails
22     a = stockdetails.stock()
23
24     open_win()
25
26 def dailyincome():
27     application.destroy()
28     login.close()
29
30     import billingdetails
31     a = billingdetails.dailyincome()
32
33     open_win()
34
35 def billingitems():
36     application.destroy()
37
38
39
40
41
42

```

STOCK DETAILS:

```

1  from tkinter import *
2  from sqlite3 import dbapi2 as sqlite
3
4  #stock details code
5
6  columns=('Item_No', 'Item_Name', 'Item_Type', 'Quantity_Remain', 'Item_Cost', 'Expiry_Date', 'Manufactured_
7
8  c=sqlite.connect("grocery.sqlite")
9  cur=c.cursor()
10
11
12
13  def autoincr():
14      ''' To auto-generate item No '''
15      cur.execute("select max(Item_No) from grocerylist")
16      incval = cur.fetchone()
17      incval = incval[0] + 1
18      # print incval
19      return incval
20
21
22  def stock():
23      ''' Stock User GUI here '''
24      global cur, c, columns, value, flag, sto, application
25
26      flag='sto'
27      value = ['']*len(columns)
28      sto=Tk()
29      sto.title('Add Stock')
30      sto.wm_iconbitmap('favicon.ico')
31      Label(sto, text='Enter a New Item to the Grocery Stock').grid(row=0, column=0, columnspan=2)
32      Label(sto, text='-'*50).grid(row=1, column=0, columnspan=2)
33
34
35      Label(sto, width=15, text=str(columns[0])+' : ', justify=LEFT).grid(row=3, column=0, sticky=W)
36      autovalue = autoincr()
37      value[0]=Entry(sto)
38      value[0].grid(row=3, column=1)
39
40      # Label(sto, text='The New Value Should be :- ' + str(autovalue)).grid(row=3, column=2)
41      value[0].insert(0, str(autovalue))

```

Expiry Check:

```

1  #Name - Dhruv Loya
2  #Email - Dhruva.loya@dcmail.ca
3  #Python 3.7
4
5  from tkinter import *
6  from sqlite3 import dbapi2 as sqlite
7  import time
8
9
10 columns=('Item_No', 'Item_Name', 'Item_Type', 'Quasntity_Remain', 'Item_Cost', 'Expiry_Date', 'Manufactured_
11
12 c=sqlite.connect("grocery.sqlite")
13 cur=c.cursor()
14
15
16
17 def expiry():
18     ''' Expiry GUI '''
19     global expirychk, expdate, c, cur, flag
20     total=0.0
21     today=str(time.localtime()[2])+ '/' +str(time.localtime()[1])+'/' +str(time.localtime()[0])
22
23     flag='expirychk'
24     groitem=[]
25     cur.execute("select * from grocerylist")
26     for i in cur:
27         groitem.append(i[1])
28     c.commit()
29     expirychk=Tk()
30     expirychk.title('Check Expiry of the Items')
31     expirychk.wm_iconbitmap('favicon.ico')
32     Label(expirychk, text='Today: '+today).grid(row=0, column=0, columnspan=3)
33     Label(expirychk, text='Its Illegal to sell expired items').grid(row=1, column=0, columnspan=3)
34     Label(expirychk, text='-'*80).grid(row=2, column=0, columnspan=3)
35     expdate=Spinbox(expirychk, values=groitem)
36     expdate.grid(row=3, column=0)
37     Button(expirychk, text='Check Expiry date', command=chkepiry).grid(row=3, column=1)
38     Label(expirychk, text='-'*80).grid(row=4, column=0, columnspan=3)
39
40     Button(expirychk, text='Main Menu', command=mainmenu).grid(row=5, column=2)
41     expirychk.mainloop()

```

BILLING CODE:

```

1  #Name - Dhruv Loya
2  #Email - Dhruva_loya@dcmail.ca
3  #Python 3.7
4
5  #for billing
6
7  from tkinter import *
8  from sqlite3 import dbapi2 as sqlite
9  import win32api
10 import win32print
11 import random
12 import time
13
14
15 columns=('Item_No', 'Item_Name', 'Item_Type', 'Quantity_Remain', 'Item_Cost', 'Expiry_Date', 'Manufactured_B'
16
17 c=sqlite.connect("grocery.sqlite")
18 cur=c.cursor()
19
20 def billingitems():
21     ''' Billing GUI '''
22     global c, cur, flag, t, name, name1, add, billingsto, names, qty, sl, qtys, n, namee, lb1
23     t=0
24
25     names=[]
26     qty=[]
27     sl=[]
28     n=[]
29     qtys=['']*10
30     cur.execute("select * from grocerylist")
31     for i in cur:
32         n.append(i[1])
33     c.commit()
34
35     flag='billingsto'
36     billingsto=Tk()
37     billingsto.title('BILLING')
38     billingsto.wm_iconbitmap('favicon.ico')
39     Label(billingsto, text="*48* Billing *48*", grid(row=0, column=0, columnspan=7, sticky='W'))

```


REFERENCES:

Shah, M. (2019, June 10). How to Apply Machine Learning (ML) in an Android App. Retrieved April 12, 2020, from <https://towardsdatascience.com/how-to-apply-machine-learning-ml-in-an-android-app-33e848c0dde6>

ChatbotNews. (2019, March 15). How To Implement Artificial Intelligence In Mobile App Development. Retrieved April 12, 2020, from <https://chatbotnewsdaily.com/how-to-implement-artificial-intelligence-in-mobile-app-development-4cf6867bb1f6>

Tutorials : TensorFlow Lite. (n.d.). Retrieved April 12, 2020, from <https://www.tensorflow.org/lite/tutorials>

TradeGecko. (2019, December 4). What is Inventory Management? 10 guides to mastering commerce. Retrieved April 12, 2020, from <https://www.tradegecko.com/inventory-management>

Udwadia, S. (2016, November 8). Grocery Store Data Set. Retrieved April 12, 2020, from <https://www.kaggle.com/shazadudwadia/supermarket>

THANK YOU