ITC

Viom Kapur

Internship Final Presentation

ITC

Mentorship Under
Mr. Sanjeev Garg

Internship Duration
1 Month (19th June 2023- 18th July 2023)

Internship Company

Undergraduate School

Viom Kapur
Bachelor's in Computer Science and Entrepreneurship/Finance

Agenda

- Introduction
- Mission
- Internship Tasks
- Projects
- What Did I Learn?
- Technical Skills
- Professional Skills
- Internship Impact
- Conclusion

MISSION

Back to Agenda Page

- Gaining valuable professional corporate experience by interning at a prestigious company like ITC.
- Collaborating closely with senior managers and project managers to understand their roles and responsibilities.
- Seizing the opportunity to learn from experienced supervisors and leveraging their guidance to acquire new skills.
- Applying these newfound skills to deliver meaningful contributions that bring value to the organization.

Internship Tasks

Projects

Acquire project assignments and thoroughly understand their requirements and use cases.

Learn

Learn new skills, tools and languages necessary to successfully execute assigned tasks and projects

Test

Utilize the acquired skills and tools to rigorously test programs and applications to meet requirements deliver the expected outcomes.

Present

Present completed projects and programs to supervisors, seeking feedback and approval. Effectively communicate the objectives and outcomes of the work done.

Back to Agenda Page

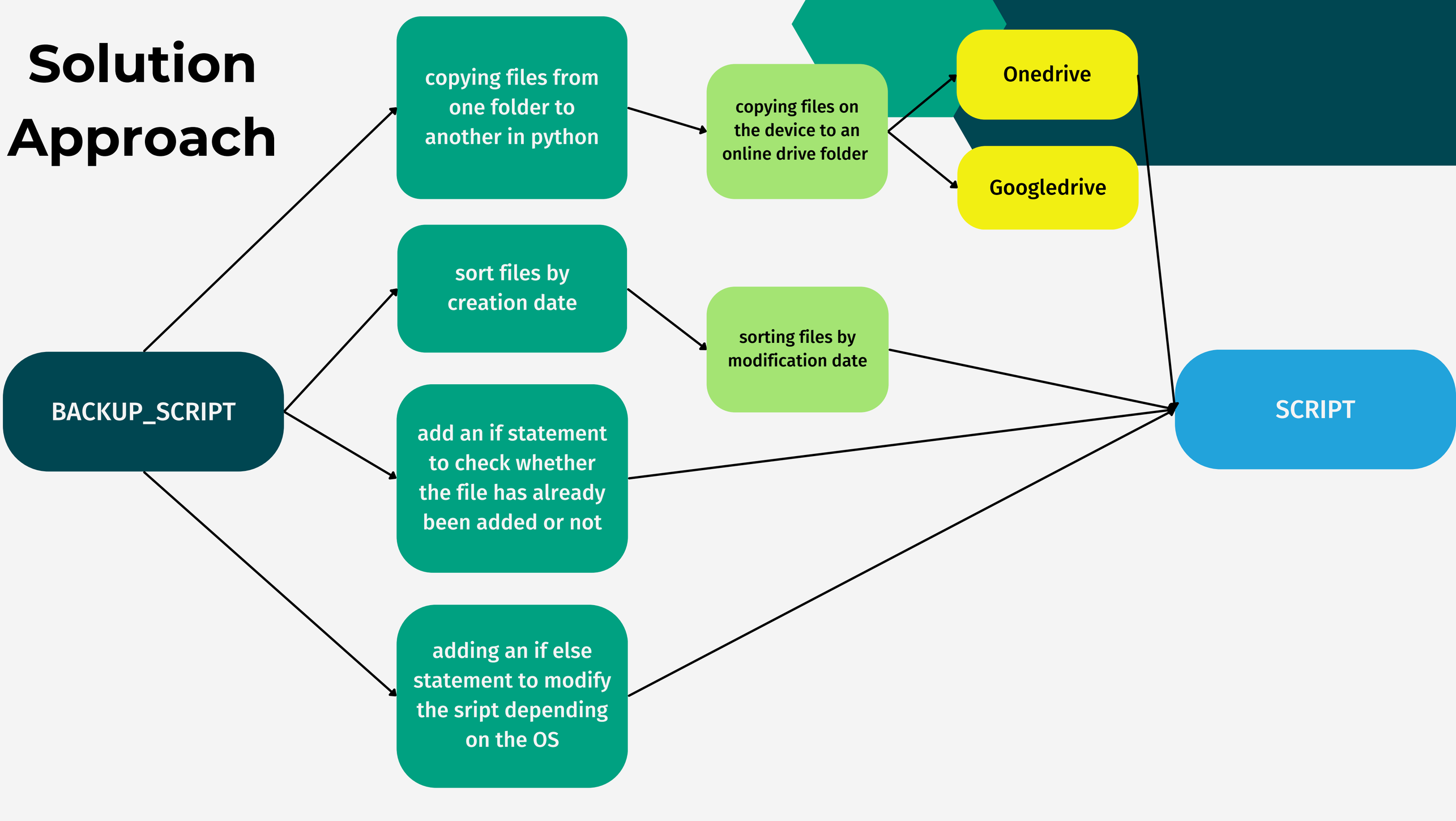
First Project

19/06/2023

Problem Statement: Develop a Python script that automates the following task: Move files from a designated folder to a Google Drive/Cloud storage folder named "backup" if they are older than one month from the current day's date.

The developed script will provide a seamless solution to automate the process of transferring older files to a designated backup location, minimizing manual effort and enhancing data management practices.

Solution Approach



Result

Through this project, I gained practical experience in automating file management using Python. By creating a script, I learned how to identify files older than a month in a specified folder and move them to a designated backup folder in Google Drive or Cloud storage. This project enhanced my Python programming skills, file handling techniques, and understanding of external API integration, enabling me to streamline repetitive tasks and improve productivity.

BACKUP_SCRIPT.PY

```python
import os
import time
import shutil
import sys

# replace with source folder path
folder = "Desktop/SEM1/calculus"

N = 30
# replace with destination folder path
destination_folder = "/Users/viomkapur/Library/CloudStorage/GoogleDrive-viomkapur@gmail.com/My Drive/backup"

os.chdir(os.path.join(os.getcwd(), folder))

list_of_files = os.listdir()

current_time = time.time()

day = 86400

if sys.platform.startswith("darwin") or sys.platform.startswith("linux"):
    # macOS
    def get_file_time(file_location):
        file_stat = os.stat(file_location)
        return file_stat.st_mtime

elif sys.platform.startswith("win"):
    # windows
    def get_file_time(file_location):
        file_stat = os.stat(file_location)
        return file_stat.st_mtime

else:
    raise NotImplementedError(f"Platform '{sys.platform}' not supported.")

for file_name in list_of_files:
    file_location = os.path.join(os.getcwd(), file_name)

    file_time = os.stat(file_location).st_mtime

    if file_time < current_time - day * N:
        copied_file_location = os.path.join(destination_folder, file_name)
        if not os.path.exists(copied_file_location):
            shutil.copy2(file_location, destination_folder)
            print(f"Copying the following: {file_name}")
        else:
            print(f"File {file_name} already exists in the folder.")

```

Use cases

1. **Time and Effort Savings:** Eliminate the need for manual inspection of file modification dates to identify outdated files for cloud backup. The program automatically identifies and copies files older than a specified duration, saving valuable time and effort.
2. **Avoiding Duplicate Uploads:** Streamline the backup process by eliminating the need to manually check the cloud folder for previously uploaded files. The program intelligently prevents duplicate uploads, ensuring that only new or modified files are copied, further saving time and effort.
3. **Data Backup:** Automatically copy files older than 30 days to a backup folder in Google Drive, ensuring data redundancy and protection against accidental loss.
4. **File Organisation:** Maintain a well-organized folder structure by moving older files to a dedicated backup folder, reducing clutter and improving file management.
5. **Version Control:** Preserve different versions of files by copying them to the backup folder, allowing easy access and retrieval of previous versions if needed.
6. **Compliance and Audit:** Comply with data retention policies or regulatory requirements by creating a backup of files that need to be retained for a specific period.
7. **Disaster Recovery:** Safeguard important files from potential system failures or data corruption by regularly copying them to a secure backup location in Google Drive.
8. **Efficient Storage Management:** Free up local storage space by moving older files to Google Drive, optimizing disk usage while retaining access to important data.

Second Project

19/06/2023

Problem Statement: Develop a Python script that automates the process of converting desired PDF files into Word file formats.

Solution Approach

The script will utilize Python libraries and modules to achieve the PDF to Word conversion efficiently.

1. Identify the desired PDF files based on specified criteria, such as file names, directories, or specific metadata.
2. Utilize a PDF processing library, such as 'pdf2docx', to extract the content from the PDF files.
3. Create a function that takes each PDF file as input and converts it into a corresponding Word file.
4. Utilize the 'pdf2docx' library's conversion capabilities to generate the Word file format.
5. Implement error handling and validation to ensure smooth execution and accurate conversion results.
6. Provide the necessary output or notifications to indicate the successful conversion of each PDF file.

By automating the PDF to Word conversion process, the script will enhance productivity and eliminate the need for manual conversion tasks. This project will provide practical experience in utilizing Python libraries, handling file formats, and automating document processing tasks, all of which are valuable skills in various professional settings.

Result

Through this project, I gained practical experience in automating the conversion of PDF files to Word file formats using Python. By creating a script that utilizes the 'pdf2docx' library, I learned how to extract text and formatting from PDFs and save them as editable Word documents. This project enhanced my understanding of file conversions, document processing, and library integration in Python. It also equipped me with a valuable skill to streamline document workflows and improve efficiency in various real-world scenarios where PDF-to-Word conversion is required.

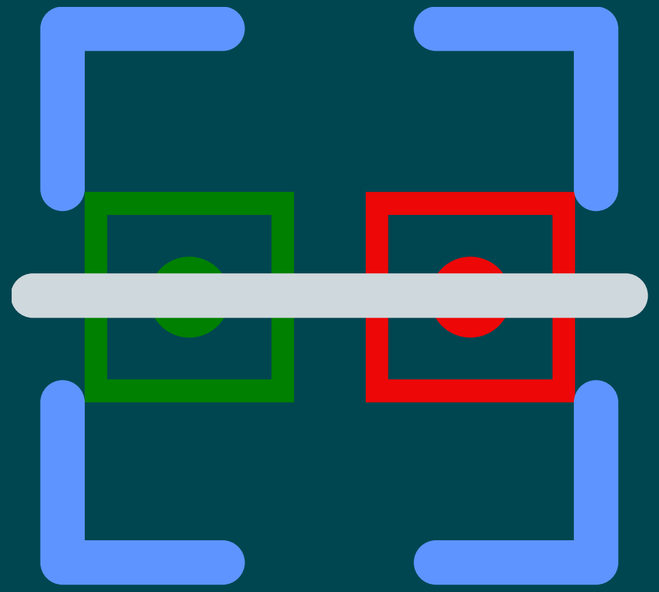
SCRIPT_PDF@DOCx.py

```python
from pdf2docx import Converter


def convert_pdf_to_word(pdf_path, output_path):
    cv = Converter(pdf_path)
    cv.convert(output_path, start=0, end=None)
    cv.close()

    print("PDF converted to Word. Saved as", output_path)


#  replace with input file path
pdf_path = "Downloads/Growth Intern JD.pdf"
# replace output file path
output_path = "Downloads/5pdftextupwork.docx"


convert_pdf_to_word(pdf_path, output_path)

```

Use cases

1. **Document Editing**: Easily edit and update PDF documents by converting them to Word format.
2. **Report Generation:** Generate customized reports by extracting data from PDF files and creating formatted Word reports.
3. **Collaboration and Review:** Facilitate team collaboration and review processes by converting PDFs to Word for easy commenting and changes.
4. **Content Extraction:** Extract specific information or data from PDF files for further processing or integration with other systems.
5. **Archiving and Documentation:** Convert historical or legacy PDF documents to Word for better preservation and accessibility.
6. **Template Creation:** Create customizable templates by converting existing PDF templates to Word format.

Third Project



Problem Statement: Develop a custom object detection model and program that can analyze a JPEG file of a menu containi*ng V-labels displaying non-veg and veg symbols. The program should be capable of detecting these symbols, determining whether they represent a veg or non-veg dish, and extracting the corresponding dish names next to the symbols as output.*

Solution Approach

Starting With Pre-trained Object Detection

Technology Used

- YOLO (You Only Look Once) in conjunction with TensorFlow.
- YOLO: A cutting-edge object detection algorithm that analyzes the entire image at once, dividing it into a grid to predict bounding boxes and class probabilities.
- TensorFlow: A powerful deep learning framework facilitating model construction, training, and deployment.

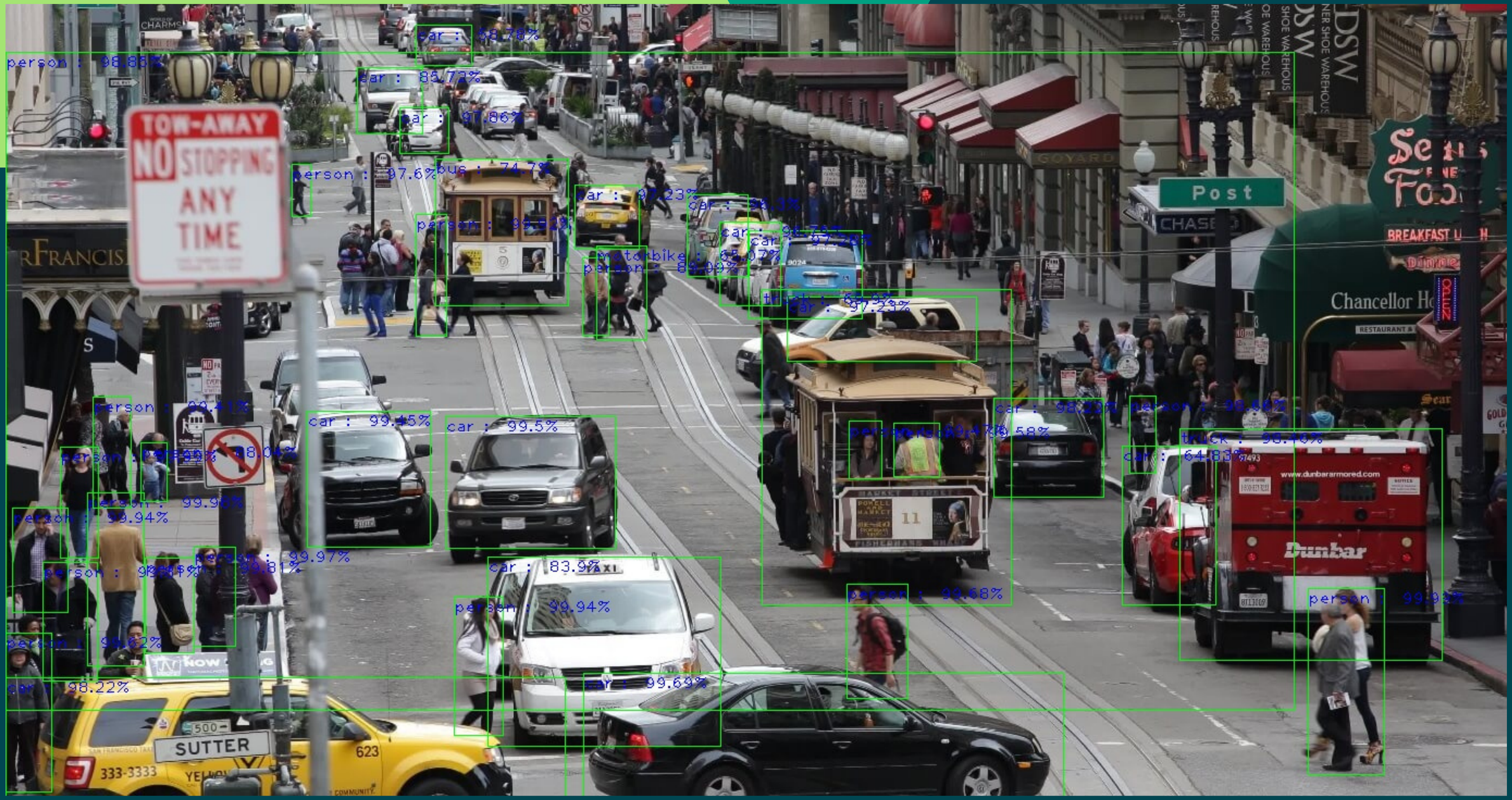
Title

```python
from imageai.Detection import ObjectDetection
import os

execution_path = os.getcwd()

detector = ObjectDetection()
detector.setModelTypeAsYOLOv3()
detector.setModelPath(
    os.path.join(
        execution_path,
"Desktop/INTERN/ITC/AITRAIN/Object_Recognition3/Model/yolov3.pt"
    )
)
detector.loadModel()
detections = detector.detectObjectsFromImage(
    input_image=os.path.join(
        execution_path,

"Desktop/INTERN/ITC/AITRAIN/Object_Recognition3/Input/sample_in.jpeg",
    ),
    output_image_path=os.path.join(
        execution_path,

"Desktop/INTERN/ITC/AITRAIN/Object_Recognition3/Output/output11_img.jpg",
    ),
    minimum_percentage_probability=30,
)

for eachObject in detections:
    print(
        eachObject["name"],
        " : ",
        eachObject["percentage_probability"],
        " : ",
        eachObject["box_points"],
    )
    print("--------------------------------")

```

To tackle the given problem statement, we will adopt a solution approach that leverages pre-trained object detection models, specifically utilizing YOLO (You Only Look Once) in conjunction with TensorFlow.

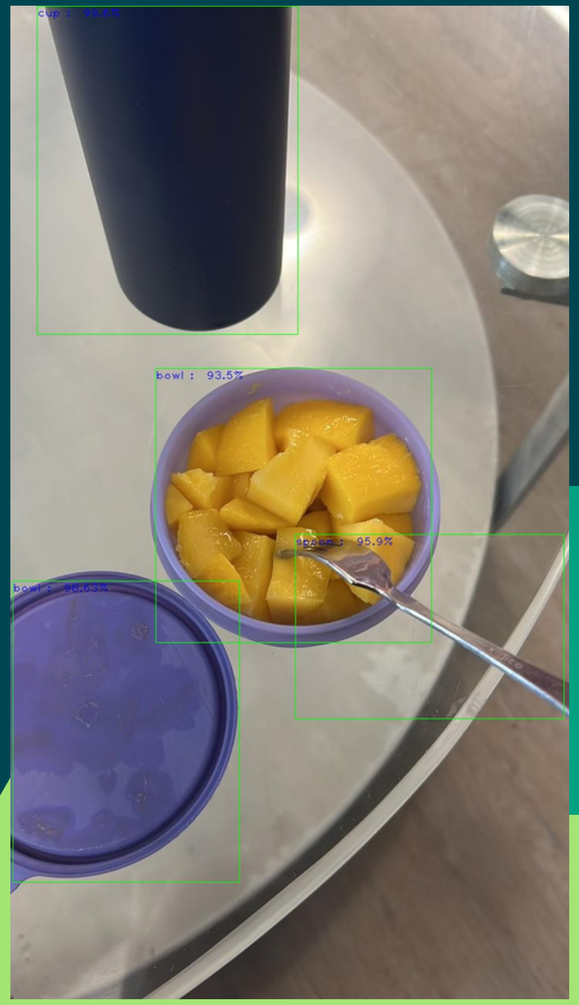
1. Pre-trained Object Detection Model:
   - Utilize a pre-trained YOLO model, which is a state-of-the-art object detection algorithm.
   - YOLO analyzes the entire image at once and divides it into a grid to predict bounding boxes and class probabilities for detected objects.
   - This approach provides real-time object detection capabilities with high accuracy.
2. TensorFlow Deep Learning Framework:
   - TensorFlow, a powerful and widely used deep learning framework, will be employed in combination with YOLO for seamless integration and model implementation.
   - TensorFlow offers a comprehensive set of tools and resources for building, training, and deploying deep learning models, including object detection models.

Input Given to the program


TOW-AWAY
NO STOPPING
ANY
TIME
Post
SUTTER
Dunbar
www.dunbararmored.com
TAXI
person : 98.85%
car : 68.78%
car : 85.73%
car : 97.86%
person : 97.6%
bus : 74.7%
car : 97.23%
car : 97.23%
motorbike :
person :
car : 99.45%
car : 99.5%
person : 99.41%
person : 99.96%
person : 99.94%
person : 99.97%
car : 83.9%
person : 99.94%
car : 99.69%
person : 98.68%
car : 98.22%
person : 98.62%
car : 98.22%
truck : 98.46%
person : 99.9%

This pretrained model has been trained to detect a wide range of everyday objects with remarkable accuracy. Some of the objects that it can detect include:

- person
- bicycle
- car
- motorcycle
- airplane
- bus
- train
- truck
- boat
- traffic light
- fire hydrant
- stop sign
- parking meter
- bench
- bird
- cat
- dog



These are just a few examples of the many objects that the pretrained model can accurately identify. Its versatility and high accuracy make it suitable for a variety of object detection tasks in different domains.

Through the use of the MySQL Connector in Python, I successfully established a connection to a MySQL database and learned how to manage databases through this integration. This achievement allowed me to effectively handle and add information to a database server using Python as the primary tool. The ability to interact with databases programmatically was a key objective of this endeavor, and it equips me with valuable skills for efficient data management and manipulation in real-world scenarios.

Title

```python
from imageai.Detection import ObjectDetection
import os
import mysql.connector

execution_path = os.getcwd()

detector = ObjectDetection()
detector.setModelTypeAsYOLOv3()
detector.setModelPath(
    os.path.join(
        execution_path,
"Desktop/INTERN/ITC/AITRAIN/Object_Recognition3/Model/yolov3.pt"
    )
)
detector.loadModel()
detections = detector.detectObjectsFromImage(
    input_image=os.path.join(
        execution_path,
        "Desktop/INTERN/ITC/AITRAIN/Object_Recognition3/Input/sample_in2.jpeg",
    ),
    output_image_path=os.path.join(
        execution_path,
        "Desktop/INTERN/ITC/AITRAIN/Object_Recognition3/Output/output5_img.jpg",
    ),
    minimum_percentage_probability=30,
)

mydb = mysql.connector.connect(
    host="localhost", user="root", password="", database="object_detection2"
)

mycursor = mydb.cursor()
mycursor.execute("DROP TABLE IF EXISTS objects")
mycursor.execute(
    "CREATE TABLE objects (id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(255),
probability FLOAT, box_points VARCHAR(255))"
)

for eachObject in detections:
    name = eachObject["name"]
    probability = eachObject["percentage_probability"]
    box_points = str(eachObject["box_points"])

    sql = "INSERT INTO objects (name, probability, box_points) VALUES (%s, %s, %s)"
    val = (name, probability, box_points)
    mycursor.execute(sql, val)

    mydb.commit()

print("Data inserted successfully!")

```

Custom Object Detection

Custom object detection is a vital computer vision task that involves training models to identify and locate specific objects within images or videos. Unlike generic object detection models, which can recognize commonly encountered objects, custom object detection enables the detection of objects specific to a particular domain or application. By training a model on a custom dataset with annotated object instances, it can learn to accurately identify and localize these specific objects. Custom object detection has wide-ranging applications, including surveillance systems, autonomous vehicles, quality control, and personalised image analysis. This powerful technology empowers businesses and researchers to tackle unique challenges and extract valuable insights from visual data.

Data Collection and Annotation:

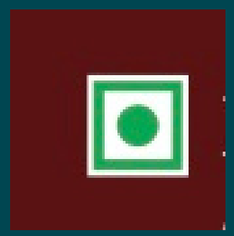
ANNOTATION

An AI model, much like a child, requires guidance and instruction to harness its capabilities effectively. In the case of custom object detection, the AI model was provided with a curated dataset comprising images containing the objects of interest, namely veg and non-veg labels.

To enable the model to understand and locate these objects within the image landscape, annotation files were meticulously created. These annotations served as a guide, teaching the AI model about the object's attributes and facilitating its learning process. This meticulous data collection and annotation process played a pivotal role in training the AI model, equipping it with the ability to accurately detect and classify veg and non-veg labels within a variety of image contexts.

annotation.txt

```
<annotation>
    <folder>images</folder>
    <filename>nonvegtrain15.jpg</filename>
    <size>
        <width>796</width>
        <height>436</height>
        <depth>3</depth>
    </size>
    <object>
        <name>NONVEG</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <occluded>0</occluded>
        <difficult>0</difficult>
        <bndbox>
            <xmin>99.875626</xmin>
            <ymin>252.422012</ymin>
            <xmax>138.826630</xmax>
            <ymax>291.332581</ymax>
        </bndbox>
    </object>
    <object>
        <name>NONVEG</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <occluded>0</occluded>
        <difficult>0</difficult>
        <bndbox>
            <xmin>127.840446</xmin>
            <ymin>39.910542</ymin>
            <xmax>166.791458</xmax>
            <ymax>78.821106</ymax>
        </bndbox>
    </object>
</annotation>

```



Model Training:

training.py

```python
import numpy as np
from imageai.Detection.Custom import DetectionModelTrainer

trainer = DetectionModelTrainer()
trainer.setModelTypeAsYOLOv3()
trainer.setDataDirectory(data_directory="Desktop/INTERN/ITC/AITRAIN/vegnonveg")
trainer.setTrainConfig(
    object_names_array=["NONVEG", "VEG"],
    batch_size=24,
    num_experiments=190,
    train_from_pretrained_model="Desktop/pretrained-yolov3.h5",
)

# Define np.float64 explicitly
np.float64

trainer.trainModel()

```

OUTPUT

output

```
/usr/local/bin/python3 /Users/viomkapur/Desktop/INTERN/ITC/AITRAIN/checking.py
Generating anchor boxes for training images and annotation...
not well-formed (invalid token): line 1, column 0
Ignore this bad annotation: Desktop/INTERN/ITC/AITRAIN/vegnonveg/train/annotations/.DS_Store
Average IOU for 9 anchors: 0.84
Anchor Boxes generated.
Detection configuration saved in  Desktop/INTERN/ITC/AITRAIN/vegnonveg/json/detection_config.json
not well-formed (invalid token): line 1, column 0
Ignore this bad annotation: Desktop/INTERN/ITC/AITRAIN/vegnonveg/validation/annotations/.DS_Store
Evaluating over 8 samples taken from Desktop/INTERN/ITC/AITRAIN/vegnonveg/validation
Training over 26 samples  given at Desktop/INTERN/ITC/AITRAIN/vegnonveg/train
Training on:    ['NONVEG', 'VEG']
Training with Batch Size:  24
Number of Training Samples:  26
Number of Validation Samples:  8
Number of Experiments:  100
Training with transfer learning from pretrained Model
WARNING:absl:At this time, the v2.11+ optimizer `tf.keras.optimizers.Adam` runs slowly on M1/M2
Macs, please use the legacy Keras optimizer instead, located at `tf.keras.optimizers.legacy.Adam`.
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy
optimizer, e.g.,tf.keras.optimizers.legacy.Adam.
WARNING:absl:There is a known slowdown when using v2.11+ Keras optimizers on M1/M2 Macs. Falling
back to the legacy Keras optimizer, i.e., `tf.keras.optimizers.legacy.Adam`.
WARNING:tensorflow:`period` argument is deprecated. Please use `save_freq` to specify the
frequency in number of batches seen.
WARNING:tensorflow:`period` argument is deprecated. Please use `save_freq` to specify the
frequency in number of batches seen.
WARNING:tensorflow:`epsilon` argument is deprecated and will be removed, use `min_delta` instead.
WARNING:tensorflow:`epsilon` argument is deprecated and will be removed, use `min_delta` instead.
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-
packages/imageai/Detection/Custom/__init__.py:308: UserWarning: `Model.fit_generator` is
deprecated and will be removed in a future version. Please use `Model.fit`, which supports
generators.
  train_model.fit_generator(
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-
packages/tensorflow/python/data/ops/structured_function.py:265: UserWarning: Even though the
`tf.config.experimental_run_functions_eagerly` option is set, this option does not apply to
tf.data functions. To force eager execution of tf.data functions, please use
`tf.data.experimental.enable_debug_mode()`.
  warnings.warn(
Epoch 1/100
 1/16 [>.............................] - ETA: 7:27 - loss: 198.4807 - yolo_layer_loss: 47.2792 -
yolo_layer_1_loss: 104.5497  2/16 [==>...........................] - ETA: 6:01 - loss: 184.9699 -
yolo_layer_loss: 60.2276 - yolo_layer_1_loss: 95.6275 - 3/16 [====>.........................] -
ETA: 5:12 - loss: nan - yolo_layer_loss: nan - yolo_layer_1_loss: nan - yolo_layer_2 4/16
[======>.....................] - ETA: 4:40 - loss: nan - yolo_layer_loss: nan -
yolo_layer_1_loss: nan - yolo_layer_2 5/16 [========>.....................] - ETA: 4:14 - loss:
nan - yolo_layer_loss: nan - yolo_layer_1_loss: nan - yolo_layer_2_loss: nan
```

Through 190 training cycles with a batch size of 24 images and annotations, and a cumulative duration of over 60 hours, several custom trained models were developed. Each successive model exhibited improved efficiency, as the training process focused on refining and optimizing the model's object detection capabilities.

Program Development:

custom_object_detection.py

```python
import numpy as np
from imageai.Detection.Custom import CustomObjectDetection


np.nan_to_num(0.0)


threshold_percentage = 35.0

detector = CustomObjectDetection()
detector.setModelTypeAsYOLOv3()
detector.setModelPath("Desktop/IMPMODEL/detection_model-ex-043--loss-0030.036.h5")
detector.setJsonPath("Desktop/INTERN/ITC/AITRAIN/vegnonveg/json/detection_config.json")
detector.loadModel()
detections = detector.detectObjectsFromImage(
    input_image="Downloads/restaurant-666054-menu-menu21627901408928.jpg",
    output_image_path="Desktop/itctest/ouput/output-detected57.jpg",
)
for detection in detections:
    name = detection["name"]
    probability = detection["percentage_probability"]
    box_points = detection["box_points"]


    if np.isnan(probability):
        probability = 0.0

    if np.isnan(box_points).any():
        box_points = [0, 0, 0, 0]


    if probability >= threshold_percentage:
        print(name, ":", probability, ":", box_points)

```

output

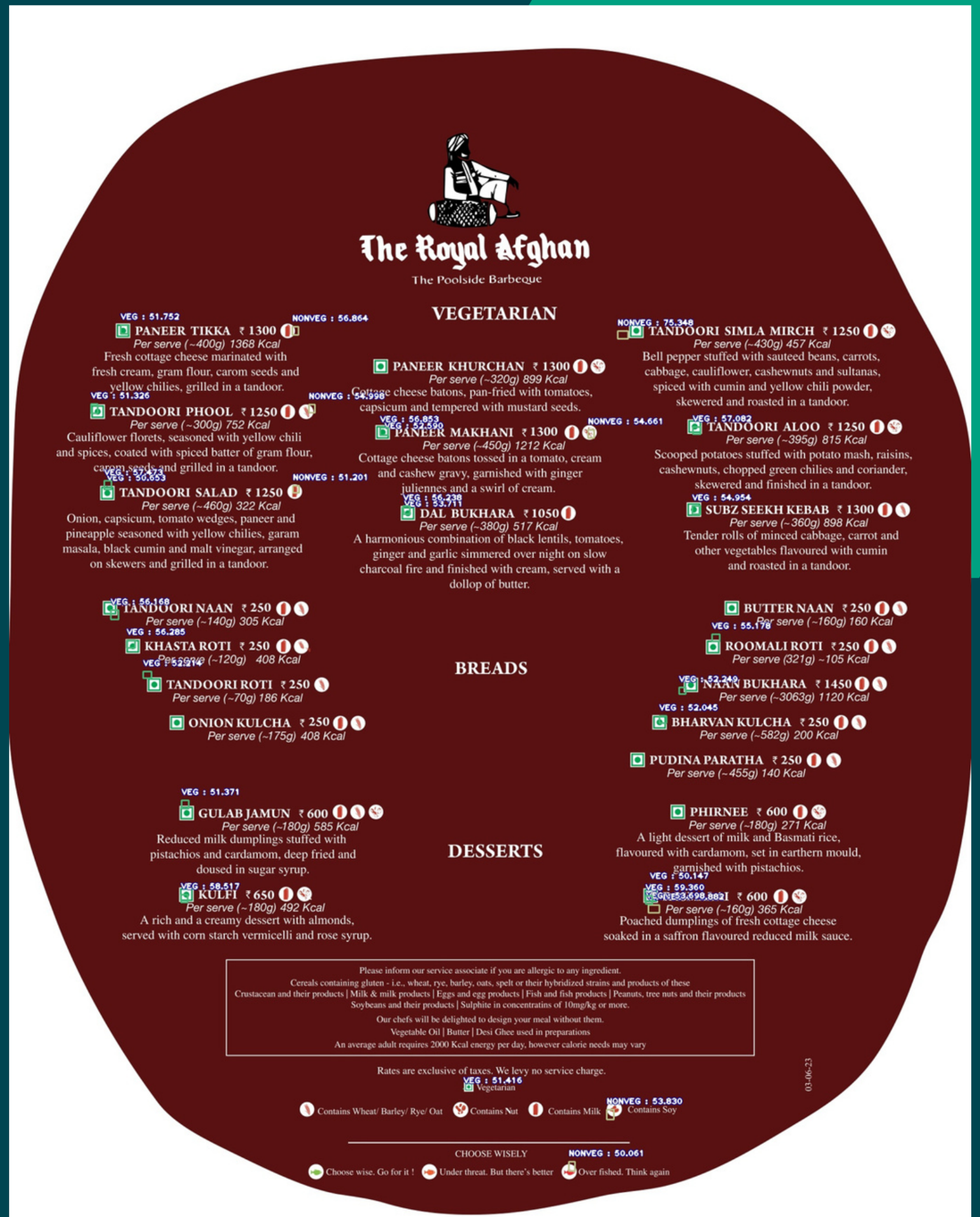
```
LABEL : Probability : [box points, box points]
```

output

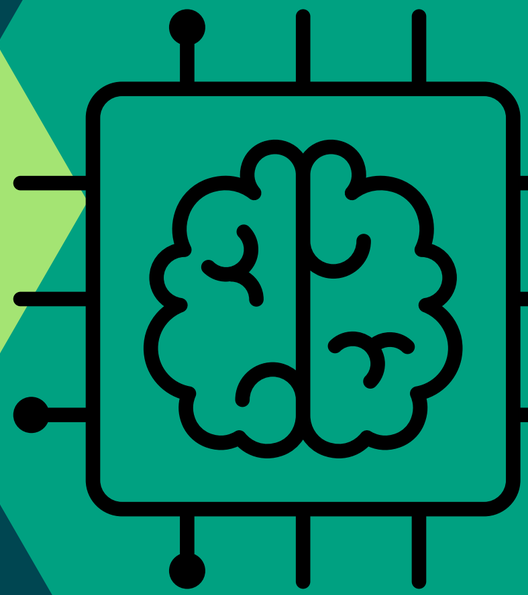
```
NONVEG : 50.14992356300354 : [1128, 749, 1140, 764]
VEG : 52.01563239097595 : [1125, 769, 1136, 782]
VEG : 52.692633867263794 : [237, 910, 249, 925]
NONVEG : 50.785237550735474 : [654, 914, 665, 929]
VEG : 51.15475058555603 : [1227, 928, 1227, 928]
NONVEG : 52.59876251220703 : [713, 1016, 724, 1031]
VEG : 53.20688486099243 : [1265, 1059, 1277, 1074]
VEG : 51.43154859542847 : [219, 1359, 219, 1359]
NONVEG : 50.009626150131226 : [1214, 1357, 1226, 1370]
VEG : 53.236621618270874 : [358, 1505, 358, 1505]
NONVEG : 50.14917850494385 : [1121, 1506, 1133, 1519]
```

Result


The Royal Afghan
The Poolside Barbeque
VEGETARIAN
PANEER TIKKA ₹ 1300
Per serve (~400g) 1368 Kcal
Fresh cottage cheese marinated with fresh cream, gram flour, carom seeds and yellow chilies, grilled in a tandoor.
TANDOORI PHOOL ₹ 1250
Per serve (~300g) 752 Kcal
Cauliflower florets, seasoned with yellow chili and spices, coated with spiced batter of gram flour, carom seeds and grilled in a tandoor.
TANDOORI SALAD ₹ 1250
Per serve (~460g) 322 Kcal
Onion, capsicum, tomato wedges, paneer and pineapple seasoned with yellow chilies, garam masala, black cumin and malt vinegar, arranged on skewers and grilled in a tandoor.
PANEER KHURCHAN ₹ 1300
Per serve (~320g) 899 Kcal
Cottage cheese batons, pan-fried with tomatoes, capsicum and tempered with mustard seeds.
PANEER MAKHANI ₹ 1300
Per serve (~450g) 1212 Kcal
Cottage cheese batons tossed in a tomato, cream and cashew gravy, garnished with ginger juliennes and a swirl of cream.
DAL BUKHARA ₹ 1050
Per serve (~380g) 517 Kcal
A harmonious combination of black lentils, tomatoes, ginger and garlic simmered over night on slow charcoal fire and finished with cream, served with a dollop of butter.
TANDOORI SIMLA MIRCH ₹ 1250
Per serve (~430g) 457 Kcal
Bell pepper stuffed with sauteed beans, carrots, cabbage, cauliflower, cashewnuts and sultanas, spiced with cumin and yellow chili powder, skewered and roasted in a tandoor.
TANDOORI ALOO ₹ 1250
Per serve (~395g) 815 Kcal
Scooped potatoes stuffed with potato mash, raisins, cashewnuts, chopped green chilies and coriander, skewered and finished in a tandoor.
SUBZ SEEKH KEBAB ₹ 1300
Per serve (~360g) 898 Kcal
Tender rolls of minced cabbage, carrot and other vegetables flavoured with cumin and roasted in a tandoor.
BREADS
TANDOORI NAAN ₹ 250
Per serve (~140g) 305 Kcal
KHASTA ROTI ₹ 250
Per serve (~120g) 408 Kcal
TANDOORI ROTI ₹ 250
Per serve (~70g) 186 Kcal
ONION KULCHA ₹ 250
Per serve (~175g) 408 Kcal
BUTTER NAAN ₹ 250
Per serve (~160g) 160 Kcal
ROOMALI ROTI ₹ 250
Per serve (321g) ~105 Kcal
NAAN BUKHARA ₹ 1450
Per serve (~3063g) 1120 Kcal
BHARVAN KULCHA ₹ 250
Per serve (~582g) 200 Kcal
PUDINA PARATHA ₹ 250
Per serve (~455g) 140 Kcal
DESSERTS
GULAB JAMUN ₹ 600
Per serve (~180g) 585 Kcal
Reduced milk dumplings stuffed with pistachios and cardamom, deep fried and doused in sugar syrup.
KULFI ₹ 650
Per serve (~180g) 492 Kcal
A rich and a creamy dessert with almonds, served with corn starch vermicelli and rose syrup.
PHIRNEE ₹ 600
Per serve (~180g) 271 Kcal
A light dessert of milk and Basmati rice, flavoured with cardamom, set in earthern mould, garnished with pistachios.
RASMALAI ₹ 600
Per serve (~160g) 365 Kcal
Poached dumplings of fresh cottage cheese soaked in a saffron flavoured reduced milk sauce.
Please inform our service associate if you are allergic to any ingredient.
Cereals containing gluten - i.e., wheat, rye, barley, oats, spelt or their hybridized strains and products of these
Crustacean and their products | Milk & milk products | Eggs and egg products | Fish and fish products | Peanuts, tree nuts and their products
Soybeans and their products | Sulphite in concentratins of 10mg/kg or more.
Our chefs will be delighted to design your meal without them.
Vegetable Oil | Butter | Desi Ghee used in preparations
An average adult requires 2000 Kcal energy per day, however calorie needs may vary
Rates are exclusive of taxes. We levy no service charge.
Vegetarian
Contains Wheat/ Barley/ Rye/ Oat
Contains Nut
Contains Milk
Contains Soy
CHOOSE WISELY
Choose wise. Go for it !
Under threat. But there's better
Over fished. Think again
03-06-23


The Royal Afghan
The Poolside Barbeque
VEGETARIAN
VEG : 51.752
NONVEG : 56.864
PANEER TIKKA ₹ 1300
Per serve (~400g) 1368 Kcal
Fresh cottage cheese marinated with fresh cream, gram flour, carom seeds and yellow chilies, grilled in a tandoor.
VEG : 51.326
NONVEG : 54.998
TANDOORI PHOOL ₹ 1250
Per serve (~300g) 752 Kcal
Cauliflower florets, seasoned with yellow chili and spices, coated with spiced batter of gram flour, carom seeds and grilled in a tandoor.
VEG : 57.473
VEG : 50.653
NONVEG : 51.201
TANDOORI SALAD ₹ 1250
Per serve (~460g) 322 Kcal
Onion, capsicum, tomato wedges, paneer and pineapple seasoned with yellow chilies, garam masala, black cumin and malt vinegar, arranged on skewers and grilled in a tandoor.
PANEER KHURCHAN ₹ 1300
Per serve (~320g) 899 Kcal
Cottage cheese batons, pan-fried with tomatoes, capsicum and tempered with mustard seeds.
VEG : 56.853
VEG : 52.499
NONVEG : 54.661
PANEER MAKHANI ₹ 1300
Per serve (~450g) 1212 Kcal
Cottage cheese batons tossed in a tomato, cream and cashew gravy, garnished with ginger juliennes and a swirl of cream.
VEG : 56.236
VEG : 53.711
DAL BUKHARA ₹ 1050
Per serve (~380g) 517 Kcal
A harmonious combination of black lentils, tomatoes, ginger and garlic simmered over night on slow charcoal fire and finished with cream, served with a dollop of butter.
NONVEG : 75.348
TANDOORI SIMLA MIRCH ₹ 1250
Per serve (~430g) 457 Kcal
Bell pepper stuffed with sauteed beans, carrots, cabbage, cauliflower, cashewnuts and sultanas, spiced with cumin and yellow chili powder, skewered and roasted in a tandoor.
VEG : 57.082
TANDOORI ALOO ₹ 1250
Per serve (~395g) 815 Kcal
Scooped potatoes stuffed with potato mash, raisins, cashewnuts, chopped green chilies and coriander, skewered and finished in a tandoor.
VEG : 54.954
SUBZ SEEKH KEBAB ₹ 1300
Per serve (~360g) 898 Kcal
Tender rolls of minced cabbage, carrot and other vegetables flavoured with cumin and roasted in a tandoor.
BREADS
VEG : 56.168
TANDOORI NAAN ₹ 250
Per serve (~140g) 305 Kcal
VEG : 56.285
KHASTA ROTI ₹ 250
Per serve (~120g) 408 Kcal
VEG : 52.814
TANDOORI ROTI ₹ 250
Per serve (~70g) 186 Kcal
ONION KULCHA ₹ 250
Per serve (~175g) 408 Kcal
BUTTER NAAN ₹ 250
Per serve (~160g) 160 Kcal
VEG : 55.178
ROOMALI ROTI ₹ 250
Per serve (321g) ~105 Kcal
VEG : 52.249
NAAN BUKHARA ₹ 1450
Per serve (~3063g) 1120 Kcal
VEG : 52.045
BHARVAN KULCHA ₹ 250
Per serve (~582g) 200 Kcal
PUDINA PARATHA ₹ 250
Per serve (~455g) 140 Kcal
DESSERTS
VEG : 51.371
GULAB JAMUN ₹ 600
Per serve (~180g) 585 Kcal
Reduced milk dumplings stuffed with pistachios and cardamom, deep fried and doused in sugar syrup.
VEG : 58.517
KULFI ₹ 650
Per serve (~180g) 492 Kcal
A rich and a creamy dessert with almonds, served with corn starch vermicelli and rose syrup.
VEG : 50.147
PHIRNEE ₹ 600
Per serve (~180g) 271 Kcal
A light dessert of milk and Basmati rice, flavoured with cardamom, set in earthern mould, garnished with pistachios.
VEG : 59.360
RASMALAI ₹ 600
Per serve (~160g) 365 Kcal
Poached dumplings of fresh cottage cheese soaked in a saffron flavoured reduced milk sauce.
Please inform our service associate if you are allergic to any ingredient.
Cereals containing gluten - i.e., wheat, rye, barley, oats, spelt or their hybridized strains and products of these
Crustacean and their products | Milk & milk products | Eggs and egg products | Fish and fish products | Peanuts, tree nuts and their products
Soybeans and their products | Sulphite in concentratins of 10mg/kg or more.
Our chefs will be delighted to design your meal without them.
Vegetable Oil | Butter | Desi Ghee used in preparations
An average adult requires 2000 Kcal energy per day, however calorie needs may vary
Rates are exclusive of taxes. We levy no service charge.
VEG : 51.416
Vegetarian
Contains Wheat/ Barley/ Rye/ Oat
Contains Nut
Contains Milk
NONVEG : 53.830
Contains Soy
CHOOSE WISELY
NONVEG : 50.061
Choose wise. Go for it !
Under threat. But there's better
Over fished. Think again
03-06-23

Model Performance and Future Enhancements



Based on the evaluation results, the model showcases an impressive accuracy rate of approximately 75%, successfully detecting 18 out of the 24 objects present in the image. This achievement is a testament to the effectiveness of the training process and the model's capability to accurately identify and locate objects. However, to further enhance its performance, future iterations can involve increasing the number of training cycles and expanding the dataset with more diverse samples. This iterative approach will enable the model to continuously update its knowledge and improve its efficiency, ultimately leading to even higher accuracy rates and more robust object detection capabilities.

Output Generation:

The item names were extracted from the menu using OCR with specific cropping coordinates and dimensions based on the largest expected size of 364x34 pixels. Unwanted text, such as prices and numbers, was filtered out from the OCR output. To align the extracted item names with the results of the v-label scans, a constant distance of 9 pixels on the x-axis between the v-labels and the food item text was maintained. This meticulous approach ensured the production of concise output that exclusively contained the desired menu item names, effectively meeting the objectives of the project.

menu_scan.py

```python
import numpy as np
from PIL import Image
import pytesseract
from imageai.Detection.Custom import CustomObjectDetection


np.nan_to_num(0.0)


pytesseract.pytesseract.tesseract_cmd = "/opt/homebrew/bin/tesseract"

detector = CustomObjectDetection()
detector.setModelTypeAsYOLOv3()
# Trained model file path
detector.setModelPath("Desktop/IMPMODEL/detection_model-ex-043--loss-0030.036.h5")
detector.setJsonPath("Desktop/INTERN/ITC/AITRAIN/vegnonveg/json/detection_config.json")
detector.loadModel()

detections = detector.detectObjectsFromImage(
    # Input image path
    input_image="Desktop/itctest/input/ITCWindsor-RoyalAfghan-Food_page-0002.jpg",
    # Output image path
    output_image_path="Desktop/itctest/output/output-detected59.jpg",
)

def perform_ocr(image):
    # Perform OCR using your chosen OCR library or method
    text = pytesseract.image_to_string(image)

    # Remove numbers from the OCR output
    text = re.sub(r"\d", "", text)

    return text.strip()


def perform_ocr(image):
    # Perform OCR using Tesseract OCR
    text = pytesseract.image_to_string(image)
    return text


for detection in detections:
    name = detection["name"]
    probability = detection["percentage_probability"]
    box_points = detection["box_points"]

    if np.isnan(probability):
        probability = 0.0

    if np.isnan(box_points).any():
        box_points = [0, 0, 0, 0]

    # Extract food item name from the image based on the box_points
    img = Image.open("Desktop/itctest/input/ITCWindsor-RoyalAfghan-Food_page-0002.jpg")
    img_width, img_height = img.size
    x1, y1, x2, y2 = box_points

    # Adjust coordinates to ensure they are within image bounds
    x1 = max(0, x1)
    y1 = max(0, y1)
    x2 = min(img_width, x2)
    y2 = min(img_height, y2)

    # Define the cropping coordinates and dimensions for the food item name
    food_item_x1 = x2 + 9
    food_item_y1 = y1
    food_item_x2 = food_item_x1 + 363
    food_item_y2 = y1 + 34

    # Perform cropping for the food item name region
    cropped_img = img.crop((food_item_x1, food_item_y1, food_item_x2, food_item_y2))
    food_item_name = perform_ocr(cropped_img)

    print(name, ":", probability, ":", box_points, ":", food_item_name)

```

Output

As you can see in the output the text recognition is working with remakrable accuracy only allowing numbers a few times and many spelling errors and misreads are occuring due to the placmenet of the scans by the program as it is only 75% effieicnt with it' current training and is thus not placing all the v-labels correctly as you can see on the next slide.

OUTPUT

```
(base) ➜  ~ /usr/local/bin/python3 /Users/viomkapur/Desktop/INTERN/ITC/AITRAIN/menureadmodel2.py
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/tensorflow/python/data/ops/structured_function.py:265: UserWarning: Even though the `tf.config.experimental_run_functions_eagerly` option is set, this option does not apply to tf.data functions. To force eager execution of tf.data functions, please use `tf.data.experimental.enable_debug_mode()`.
  warnings.warn(
1/1 [==============================] - 0s 402ms/step
NONVEG : 75.34804940223694 : [1109, 692, 1128, 706] : TANDOORI SIMLA MIRCH

NONVEG : 73.88153076171875 : [1165, 1739, 1185, 1753] :RASMALAI Per serve (~160g) 365 Kcal

dumplings of frech cottace cheece

VEG : 51.752495765686035 : [203, 681, 216, 697] : PANEER TIKKA

NONVEG : 56.86376690864563 : [516, 684, 527, 699] :

VEG : 51.326411962509155 : [149, 825, 161, 842] : | TANDOORI PHOOL

NONVEG : 54.99773621559143 : [546, 826, 557, 841] : Capsicum and tempered with mi

VEG : 52.5896430015564 : [684, 880, 684, 880] : Per serve  Keal

VEG : 56.85284733772278 : [676, 868, 690, 882] : PANEER MAKHANI

NONVEG : 54.66083884239197 : [1055, 872, 1066, 887] :

VEG : 57.081544399261475 : [1246, 867, 1260, 880] : TANDOORI ALOO
Per serve Kcal

VEG : 50.65300464630127 : [178, 975, 178, 975] : | TANDOORI SALAD

VEG : 57.47264623641968 : [173, 965, 186, 981] : TANDOORI SALAD

NONVEG : 51.20117664337158 : [517, 974, 528, 987] : Jutiennes and a swir

VEG : 53.71135473251343 : [720, 1022, 720, 1022] : VAL DUNTIANA

VEG : 56.23829364776611 : [718, 1011, 730, 1026] : | DAL BUKHARA

VEG : 54.95402812957764 : [1245, 1011, 1260, 1026] : SUBZ SEEKH KEBAB
VEG : 56.16763234138489 : [185, 1201, 199, 1218] : Per serve (~140g) 305 Keal

VEG : 56.28488063812256 : [214, 1256, 229, 1270] : KHASTA ROTI


VEG : 55.17771244049072 : [1281, 1245, 1295, 1258] : ROOMALIROTI 7250 QO @®

VEG : 52.21419930458069 : [244, 1313, 259, 1325] : 1 TANDOORIROTI 7250 @

VEG : 52.248966693878174 : [1221, 1342, 1233, 1354] : Per: serve (~3063g) 1120 Ke cal

VEG : 52.04507112503052 : [1185, 1394, 1198, 1407] : BHARVAN KULCHA 7250

VEG : 51.3710618019104 : [314, 1548, 327, 1563] : GULAB JAMUN 7600 0O@ @

VEG : 50.14721751213074 : [1168, 1701, 1168, 1701] : ] RASMALAI 7600 Q @

VEG : 58.51690173149109 : [313, 1720, 325, 1734] : KULFI(~180q) 492 Kcal

VEG : 53.69846820831299 : [1161, 1737, 1161, 1737] : Per serve (~160g) 365 Kcal frach
 cattagde cheaca

VEG : 59.36020612716675 : [1160, 1722, 1172, 1737] : Per serve (~160q) 365 Kcal

VEG : 51.416200399398804 : [828, 2074, 828, 2074] :
NONVEG : 53.82972955703735 : [1089, 2112, 1102, 2130] :
NONVEG : 50.06126165390015 : [1020, 2207, 1031, 2224] : Over fished. Think again
```

The errors encountered do not indicate inefficiency in the code, libraries, or program. Rather, they serve as a reminder that the model would benefit from additional training cycles using a more extensive and diverse dataset.

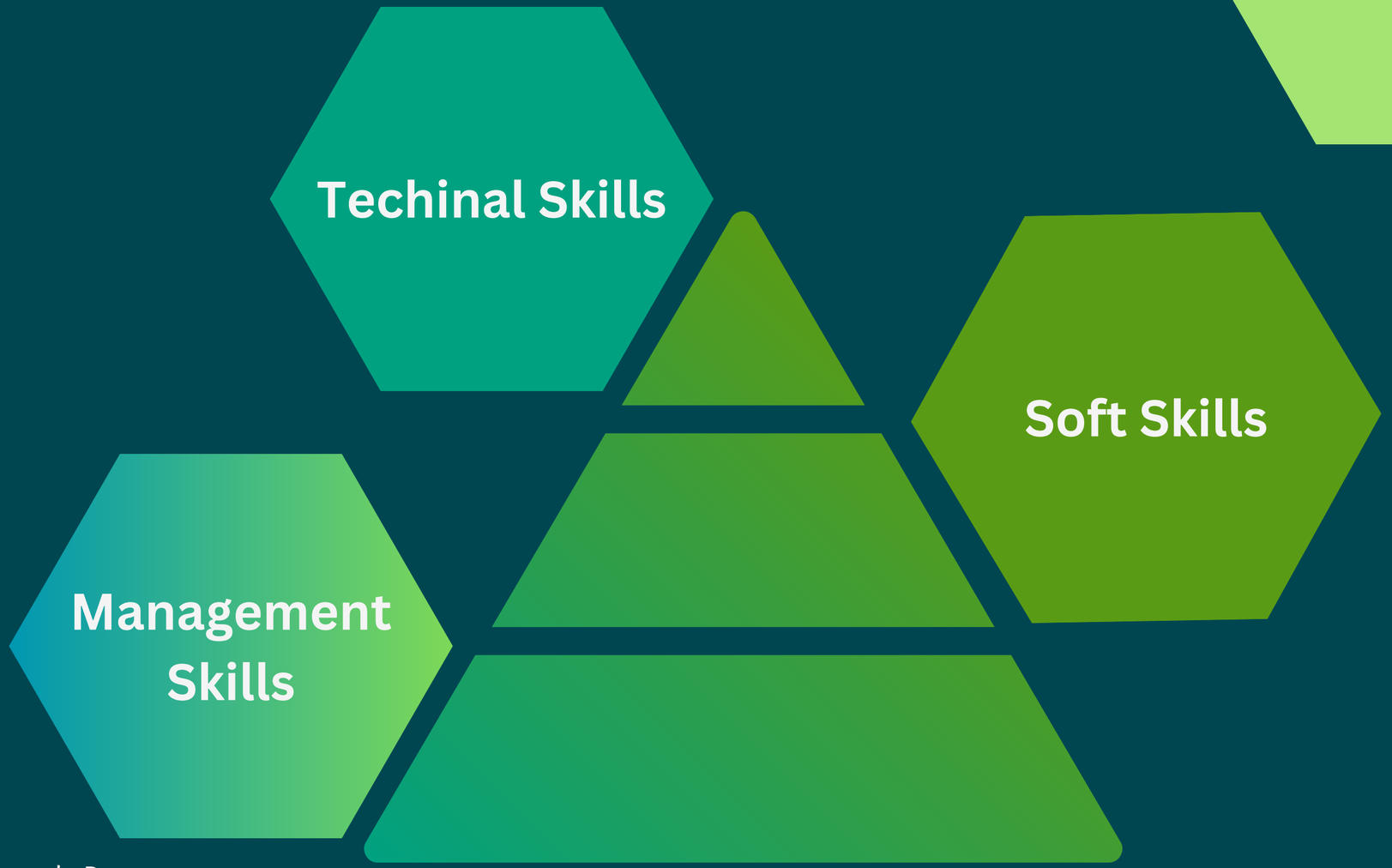
These misplacements of the labels and labelling of the fish label as the non veg label are all errors leading to the outputs "Per serve (~160g) 365 Kcal " by the OCR in the previous program as the labels are being positioned a bit under the V-labels.

The program's ability to detect and present 18 out of the 24 labels in the image, achieving a 75% success rate with limited training, highlights the efficiency of the program and the effectiveness of the utilized libraries.

Use cases

1. **Surveillance and Security Systems:** Custom object detection allows for the identification and tracking of specific objects of interest in surveillance footage, such as individuals, vehicles, or suspicious objects. This helps enhance security measures and enables proactive threat detection.
2. **Industrial Quality Control:** Custom object detection can be used to detect and classify defects or anomalies in manufacturing processes. It enables automated inspection of products, ensuring consistent quality and minimizing errors.
3. **Autonomous Vehicles:** Custom object detection plays a crucial role in enabling self-driving cars and other autonomous vehicles to detect and recognize various objects in real-time, including pedestrians, traffic signs, and other vehicles. This helps ensure safety and facilitates intelligent decision-making.
4. **Medical Imaging:** Custom object detection can aid in medical image analysis, identifying and localizing specific structures or abnormalities in scans such as tumors, organs, or anatomical landmarks. It assists in diagnosis, treatment planning, and monitoring of diseases.
5. **Retail Analytics:** Custom object detection enables the analysis of shopper behavior, tracking specific products or items within a store environment. It provides insights into customer preferences, inventory management, and optimizing store layouts for better customer experience.
6. **Document Processing:** Custom object detection can be utilized in automating document processing tasks, such as recognizing and extracting specific information from forms or invoices. This streamlines administrative workflows and enhances efficiency.

What did I learn ?



Back to Agenda Page

Technical Skills

Python

AI

SQL

API's

Machine Learning

Amazon Web Services

Modules

Custom Object Detection

Professional Skills

Soft Skills

- **Teamwork** in a professional corporate environment.
- Demonstrating **ethical behavior, maintaining confidentiality, being punctual,** and displaying a positive attitude in the workplace.
- **Adapting** to new environments, technologies, and work methodologies, and being open to feedback and constructive criticism.

Management Skills

- Project management
- **Time** management
- **Problem Solving**: Developing critical thinking abilities, identifying and analyzing problems, and proposing creative solutions.

Internship Impact

Short Term Advantages

- Enhancing verbal and written communication skills, effectively conveying ideas, active listening, and collaborating with team members.
- Demonstrating ethical behavior, maintaining confidentiality, being punctual, and displaying a positive attitude in the workplace.
- Being open to change, adjusting to evolving project requirements, and demonstrating resilience in challenging situations.

Long Term Advantages

- Gaining valuable experience in the rapidly growing field of Artificial Intelligence, equipping oneself with knowledge and skills that are increasingly in demand in various industries.
- Applying the theoretical knowledge gained in the classroom to real-world projects, understanding how concepts translate into practical solutions, and developing a deeper understanding of the subject matter.
- Obtaining insights into one's professional career path and gaining clarity on future aspirations, enabling informed decisions and planning for future growth opportunities.
- Enhanced Employability: ability to apply technical knowledge in a business context and effectively contribute to digital transformation initiatives.

Conclusion

In conclusion, this internship has been an excellent and rewarding experience. Working alongside supportive supervisors and engaging in meaningful and relevant projects has greatly contributed to my personal and professional growth. The opportunity to work on projects that align with my interests has allowed me to further develop my skills and knowledge in the field. The collaborative and conducive work environment fostered a culture of learning and skill enhancement, enabling me to expand my professional capabilities. I am grateful for the valuable skills acquired during this internship, and I am confident that they will positively influence my future career endeavors. This experience has instilled in me a strong passion for continuous learning and innovation, and I am excited about the opportunities that lie ahead. Overall, this internship has been instrumental in shaping my career trajectory, and I am grateful for the valuable lessons learned and the relationships established throughout this journey.