

# Internship Report

J Jayagowtham

B.Tech in Mechanical Engineering

Internship Dates

2nd June 2023 to 30th July 2023 – Full Time

31st July 2023 to 21st October 2023 – Part Time

Center for Non-Destructive Evaluation (CNDE) IIT

Madras Chennai -36

# Appendix

- Acknowledgment
- Information about the Internship Position
- About the Project
- Description of the Internship Experience
- Work on the Shobot Project
- Results
- Conclusion
- Links to Repositories
- Literature and References

## Acknowledgment

I want to thank Prof. Prabhu Rajagopal for giving me this internship opportunity and everyone at CNDE for their patience and assistance during my 2-month internship. Thanks to their guidance, I was able to implement an ML model hands on and

learn about the functioning of the robotic arm. This internship has been transformative, and I am thankful for the knowledge and skills gained during this time.

## **Information about the internship position**

As a project intern at CNDE, I was responsible for the object detection capabilities of the Shobot. I worked on the acquisition and creation of dataset, the ML model used to train and retrieve the final coordinates of an object after successfully identifying it.

## **About the Project**

The objective is to develop an autonomous grocery shopping assistant robot to improve the customer shopping experience with the least human interaction. The robot is structured to have the UR5 Robotic arm mounted on an Autonomous Mobile Robot (AMR) ANCER Platform. The robot performs tasks such as object identification, pick and place, and product delivery. The Shobot has a database to store and register customers to assist them in grocery shopping. The arm operates autonomously in optimized paths to provide the users with quicker responses. The Robot has integrated obstacle avoidance in its environment mapping to steer clear of the shelf's presence. The introduction of Robots to assist in pick and place activities has deeply reduced the need for human intervention and minimized the cost of human labour in retail shops.

Further developments include manufacturing the base platform in-house in IITM Workshops and building a customized SCARA arm with the intention of reducing the cost of the overall project.



Fig. 0 The Shobot

## Description of internship experience

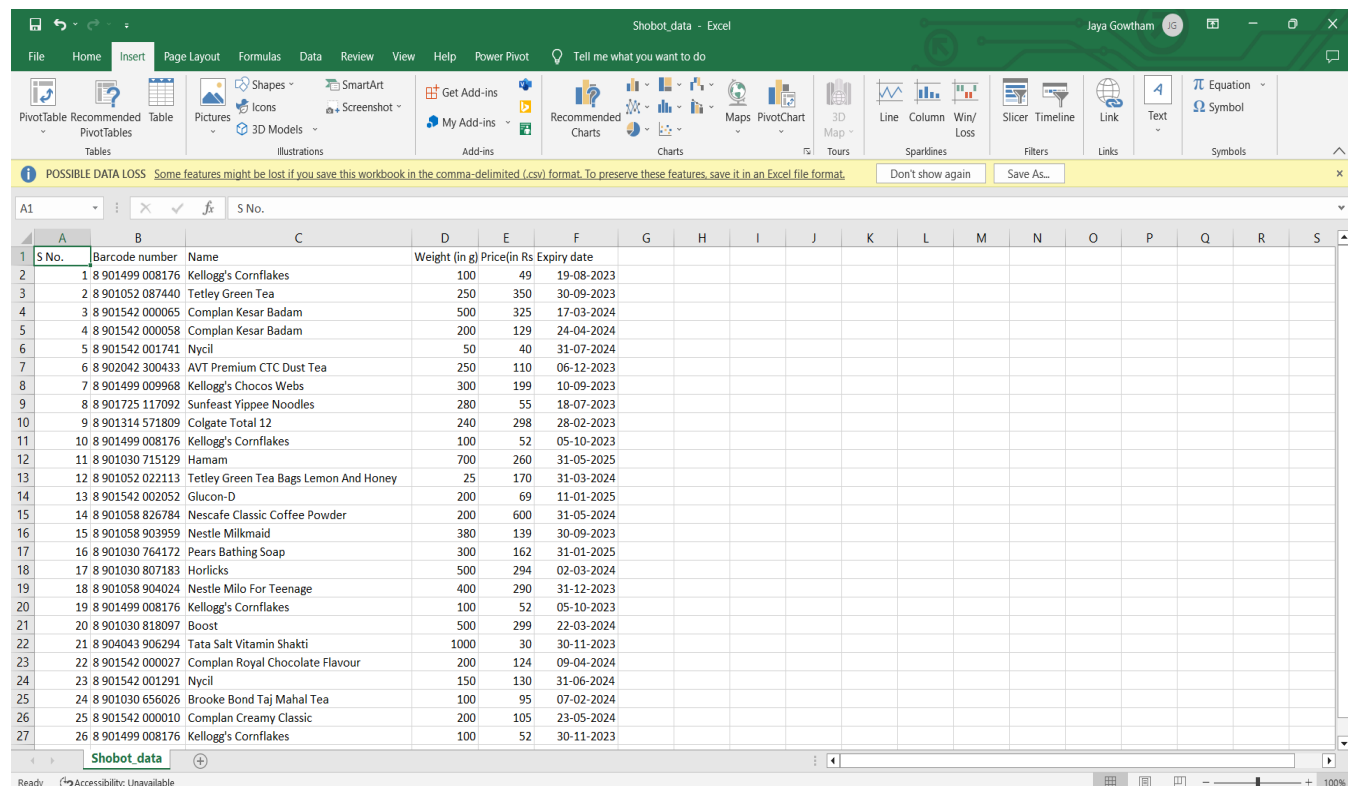
During my internship, I was tasked with creating a custom model for detecting grocery items. I worked in the well-equipped and air conditioned HPCF lab, and reported my work to the project head on a weekly basis. I acquired hands on experience of working with ML and understood the working of various state-of-the-art object detection models. I followed a structured approach, starting from defining the problem statement and progressing towards delivering a final ML-based product. I also tried to comprehend the working of the Shobot, which was initially unclear due to lack of information. The testing of arm to integrate it into my work was thrilling and exciting.

In addition to the specific tasks mentioned, I actively participated in discussions with fellow interns to collaborate and ideate for further developments in the project. These activities allowed me to broaden my understanding of current research trends and methodologies in the field. I also had the opportunity to interact with other researchers and lab members, which enhanced my teamwork and communication skills.

## Detailed Work Report

### Panel app and Python libraries

During my internship, my first task involved preparing an Excel sheet to catalogue the items within the designated arena (refer to Fig.1). This process required manual collection of data, including crucial information such as barcode numbers, product names, weights, prices, and expiry dates.



S No.	Barcode number	Name	Weight (in g)	Price(in Rs)	Expiry date
1	8 901499 008176	Kellogg's Cornflakes	100	49	19-08-2023
2	8 901052 087440	Tetley Green Tea	250	350	30-09-2023
3	8 901542 000065	Complan Kesar Badam	500	325	17-03-2024
4	8 901542 000058	Complan Kesar Badam	200	129	24-04-2024
5	8 901542 001741	Nycil	50	40	31-07-2024
6	8 902042 300433	AVT Premium CTC Dust Tea	250	110	06-12-2023
7	8 901499 009968	Kellogg's Chocos Webs	300	199	10-09-2023
8	8 901725 117092	Sunfeast Yippee Noodles	280	55	18-07-2023
9	8 901314 571809	Colgate Total 12	240	298	28-02-2023
10	8 901499 008176	Kellogg's Cornflakes	100	52	05-10-2023
11	8 901030 715129	Hamam	700	260	31-05-2025
12	8 901052 022113	Tetley Green Tea Bags Lemon And Honey	25	170	31-03-2024
13	8 901542 002052	Glucon-D	200	69	11-01-2025
14	8 901058 826784	Nescafe Classic Coffee Powder	200	600	31-05-2024
15	8 901058 903959	Nestle Milkmaid	380	139	30-09-2023
16	8 901030 764172	Pears Bathing Soap	300	162	31-01-2025
17	8 901030 807183	Horlicks	500	294	02-03-2024
18	8 901058 904024	Nestle Milo For Teenage	400	290	31-12-2023
19	8 901499 008176	Kellogg's Cornflakes	100	52	05-10-2023
20	8 901030 818097	Boost	500	299	22-03-2024
21	8 904043 906294	Tata Salt Vitamin Shakti	1000	30	30-11-2023
22	8 901542 000027	Complan Royal Chocolate Flavour	200	124	09-04-2024
23	8 901542 001291	Nycil	150	130	31-06-2024
24	8 901030 656026	Brooke Bond Taj Mahal Tea	100	95	07-02-2024
25	8 901542 000010	Complan Creamy Classic	200	105	23-05-2024
26	8 901499 008176	Kellogg's Cornflakes	100	52	30-11-2023

Fig. 1 Excel sheet of our arena's arsenal

I applied my knowledge of python to develop a dedicated panel application for the Excel sheet, as illustrated in Fig. 2. The primary objective of this application was to facilitate the retrieval of information pertaining to any item based on its barcode number. The application was linked to the previously created Excel sheet, creating a connection between the data and the user interface.

I opted for a basic template called "Bootstrap" which ensured a user-friendly design for the application. Additionally, I incorporated interactive widgets that added a dynamic dimension to the app.

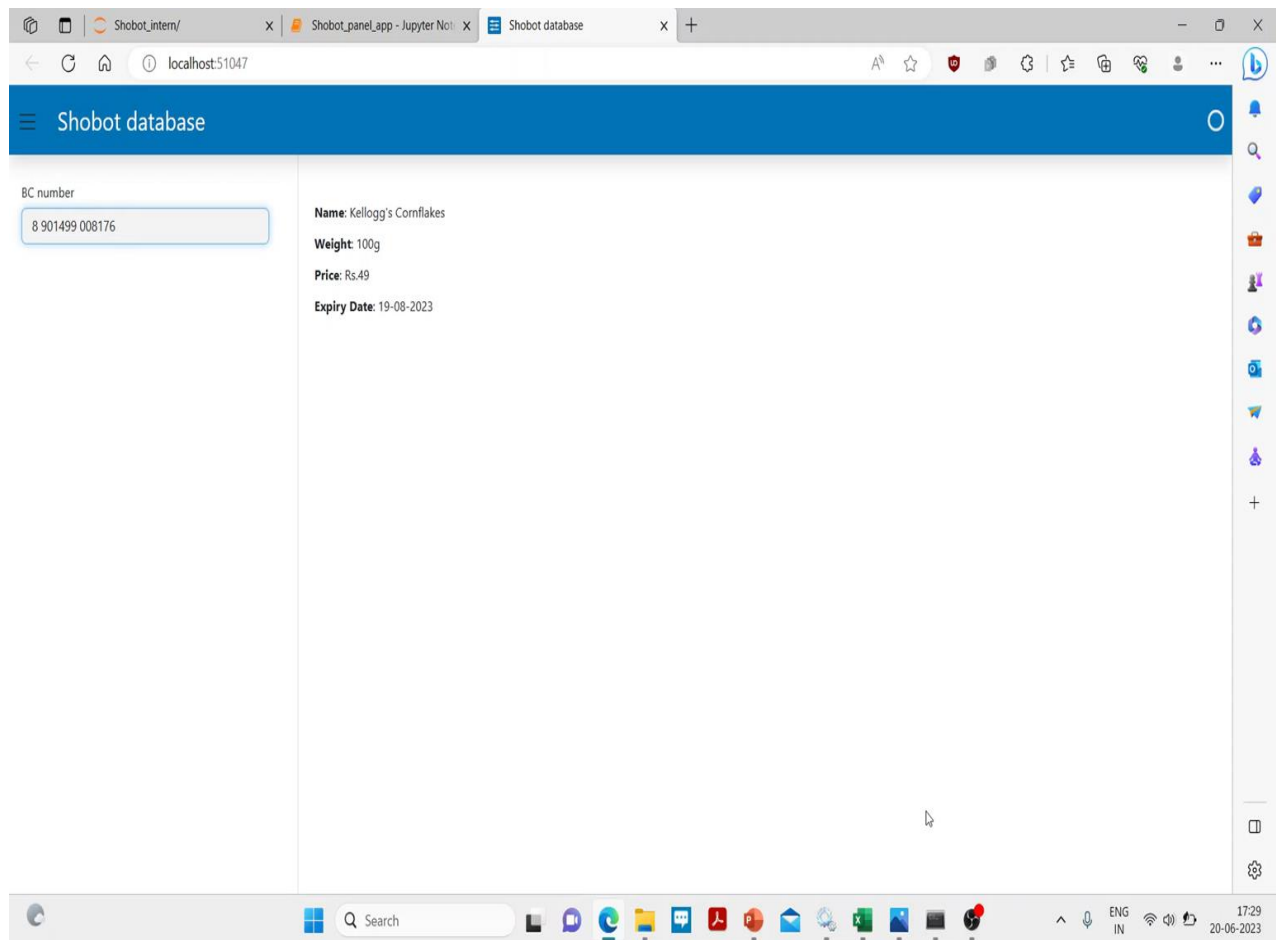
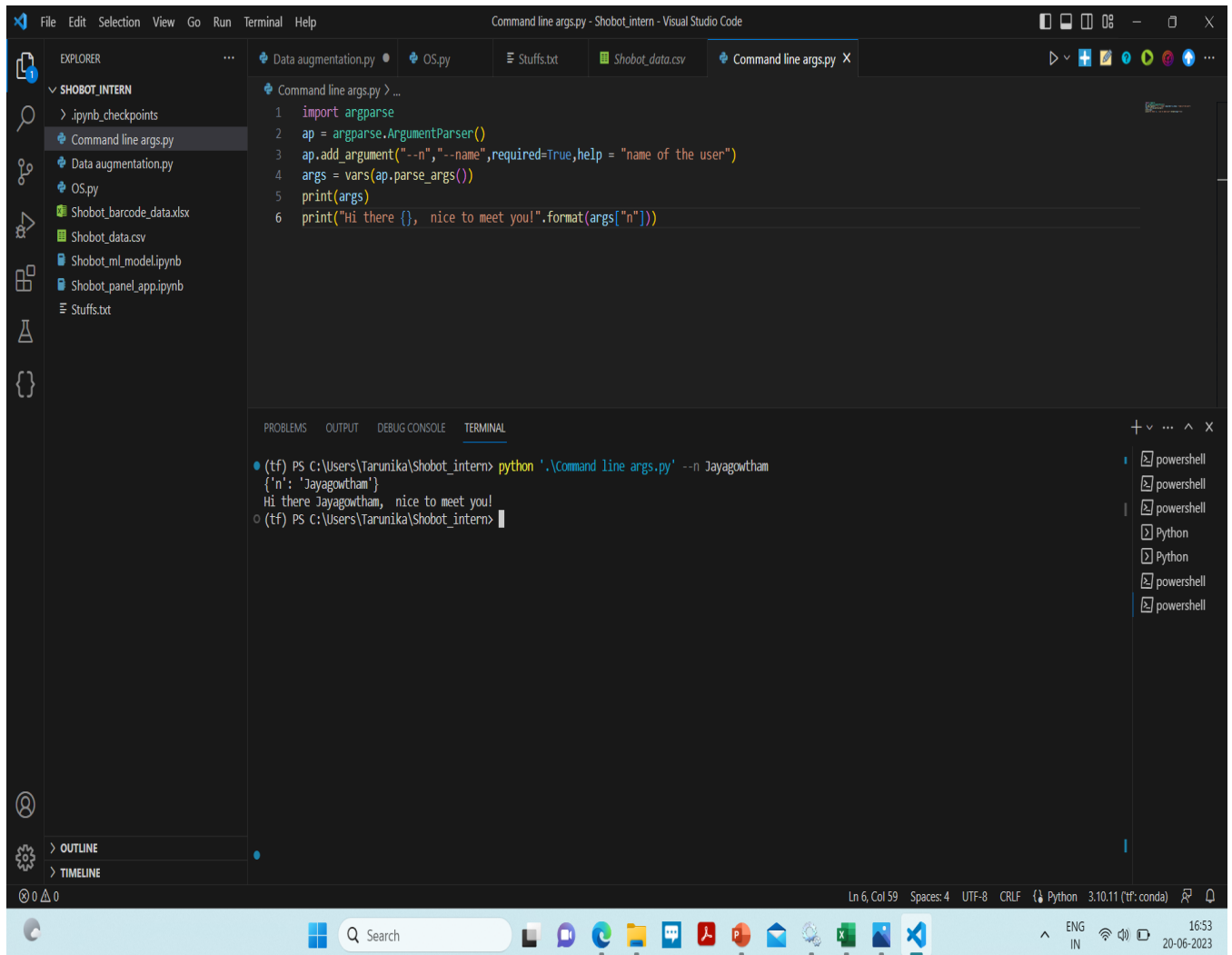


Fig. 2 Panel application

I learnt argument parsing in Python, as depicted in Fig. 3. I comprehended the significance of argument parsing in command line operations, particularly its usefulness in interpreting and collecting arguments effectively. This helped me in the context of creating image

datasets, where command line instructions are essential. I also learnt file management using the Python “os” module, as shown in Fig. 4. This learning is crucial for executing various operations within the current directory, such as file or directory creation, opening files by path, and obtaining a list of available documents. It is beneficial in the creation and storage of image datasets.



The screenshot displays the Visual Studio Code interface with a project named 'Shobot\_intern'. The Explorer sidebar on the left shows the file structure, including 'Command line args.py'. The main editor window shows the code for 'Command line args.py', which uses the 'argparse' module to handle command-line arguments. The code defines an argument parser, adds a required argument '--name', and prints the parsed arguments. The Terminal at the bottom shows the command 'python .\Command line args.py --n Jayagowtham' being executed, resulting in the output 'Hi there Jayagowtham, nice to meet you!'. The status bar at the bottom indicates the current file is 'Command line args.py' at line 6, column 59, using UTF-8 encoding and CRLF line endings.

```
1 import argparse
2 ap = argparse.ArgumentParser()
3 ap.add_argument("--n", "--name", required=True, help = "name of the user")
4 args = vars(ap.parse_args())
5 print(args)
6 print("Hi there {}, nice to meet you!".format(args["n"]))
```

```
(tf) PS C:\Users\Tarunika\Shobot_intern> python .\Command line args.py --n Jayagowtham
{'n': 'Jayagowtham'}
Hi there Jayagowtham, nice to meet you!
(tf) PS C:\Users\Tarunika\Shobot_intern>
```

Fig. 3 Argument parsing

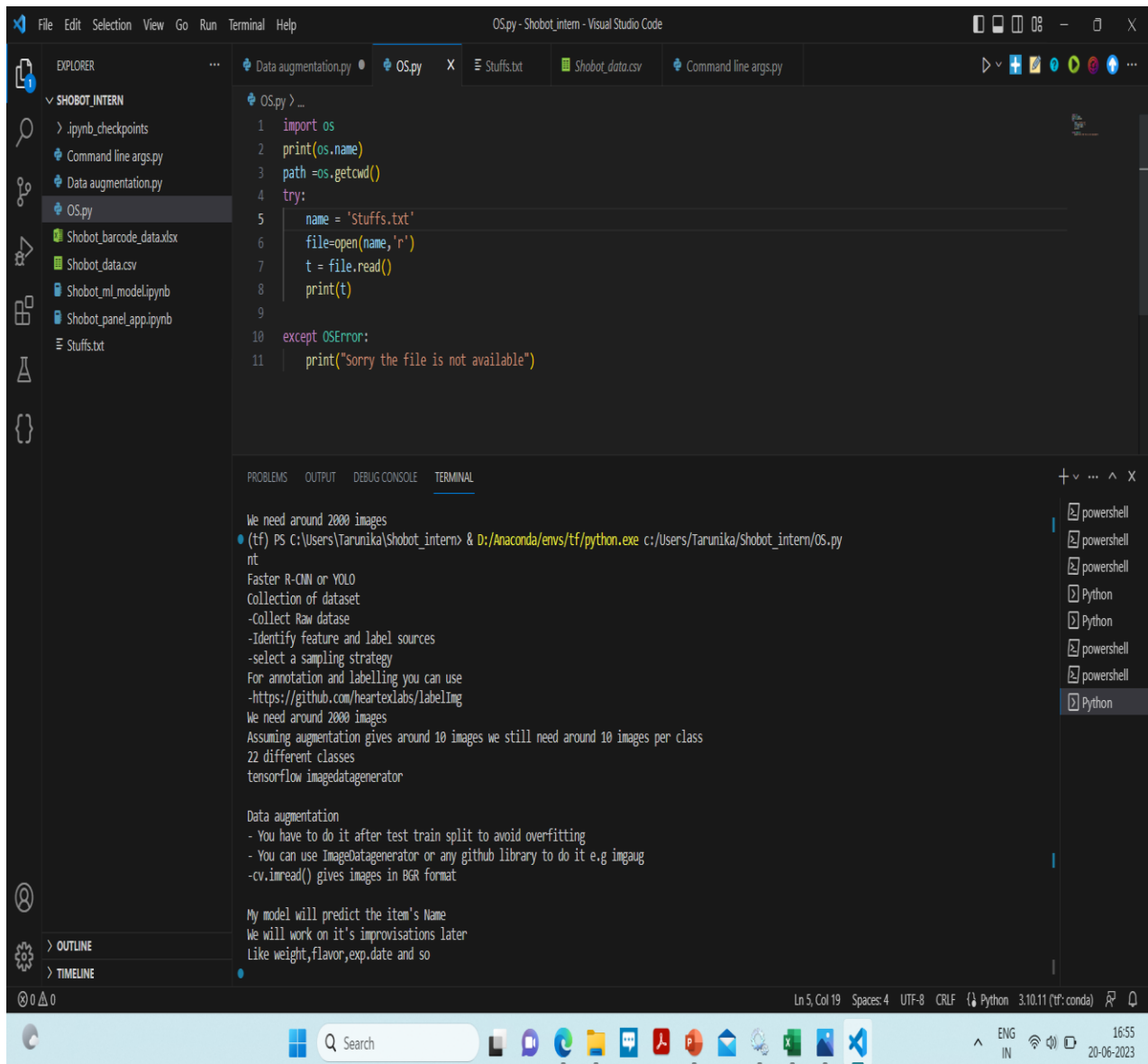


Fig. 4 OS Library

## Data Acquisition

In the process of dataset creation, a significant step was to acquire images from Google. The methodology for this is detailed in Fig. 4, Fig. 5, and Fig. 6, where a JavaScript code was employed to extract URLs as I scrolled through the search results. Subsequently, a Python script was developed to iterate through all the specified labels, downloading all the images in a single operation.



```

function simulateRightClick( element ) {
    var event1 = new MouseEvent( 'mousedown', {
        bubbles: true,
        cancelable: false,
        view: window,
        button: 2,
        buttons: 2,
        clientX: element.getBoundingClientRect().x,
        clientY: element.getBoundingClientRect().y
    } );
    element.dispatchEvent( event1 );
    var event2 = new MouseEvent( 'mouseup', {
        bubbles: true,
        cancelable: false,
        view: window,
        button: 2,
        buttons: 0,
        clientX: element.getBoundingClientRect().x,
        clientY: element.getBoundingClientRect().y
    } );
    element.dispatchEvent( event2 );
    var event3 = new MouseEvent( 'contextmenu', {
        bubbles: true,
        cancelable: false,
        view: window,
        button: 2,
        buttons: 0,
        clientX: element.getBoundingClientRect().x,
        clientY: element.getBoundingClientRect().y
    } );
    element.dispatchEvent( event3 );
}

```

Fig. 4 Function to simulate right click on a webpage

```

function getUrlParam( queryString, key ) {
    var vars = queryString.replace( /\?/, '' ).split( '&' );
    for ( let i = 0; i < vars.length; i++ ) {
        let pair = vars[ i ].split( '=' );
        if ( pair[0] == key ) {
            return pair[1];
        }
    }
    return false;
}

```

Fig. 5 Function to extract the contents of an image

```

function createDownload( contents ) {
    var hiddenElement = document.createElement( 'a' );
    hiddenElement.href = 'data:attachment/text,' + encodeURIComponent( contents );
    hiddenElement.target = '_blank';
    hiddenElement.download = 'urls.txt';
    hiddenElement.click();
}

```

Fig. 6 Function to download the image

It appeared that some images were either corrupt or unnecessary for the intended dataset. To address this issue, firstly, a Python script was implemented to identify and remove images that could not be opened by OpenCV, ensuring the efficient cleansing of the data.

```

for url in rows:
    try:
        # try to download the image
        r = requests.get(url, timeout=60)

        # save the image to disk
        p = os.path.sep.join([args["output"], "{}.jpg".format(
            str(total).zfill(8))])
        f = open(p, "wb")
        f.write(r.content)
        f.close()

        # update the counter
        print("[INFO] downloaded: {}".format(p))
        total += 1

    # handle if any exceptions are thrown during the download process
    except:
        print("[INFO] error downloading {}...skipping".format(p))

```

Fig. 7 Downloading the images

```

for imagePath in paths.list_images(args["output"]):
    # initialize if the image should be deleted or not
    delete = False

    # try to load the image
    try:
        image = cv2.imread(imagePath)

        # if the image is 'None' then we could not properly load it
        # from disk, so delete it
        if image is None:
            delete = True

        # if OpenCV cannot load the image then the image is likely
        # corrupt so we should delete it
    except:
        print("Except")
        delete = True

    # check to see if the image should be deleted
    if delete:
        print("[INFO] deleting {}".format(imagePath))
        os.remove(imagePath)

```

Fig. 8 Removing corrupt or unnecessary images

Manual intervention was also necessary to further refine the dataset. Each image was individually analysed, and the unnecessary ones were deleted. This ensured that the final dataset comprised only relevant and reliable images for subsequent analysis.





Fig. 9 Sample images

## Dataset creation and Preliminary model

The process of creating the dataset initiated with the upload of 3000+ images to the Roboflow website, where, in collaboration with a fellow intern, over 500 images underwent manual annotation for improved accuracy.

Since the process was deemed time intensive, a decision was made to proceed with a subset of approximately 400 images for the initial model, focusing on four classes - Horlicks, Lays, Colgate, and Hamam (refer to Fig. 10). The model training, executed on the Roboflow platform, produced an impressive accuracy of 81.8 percent for the specified classes. Training metrics, visualized in Fig. 11, showcased a decreasing loss and increasing accuracy, but showed some abruptness due to the limited number of input images.

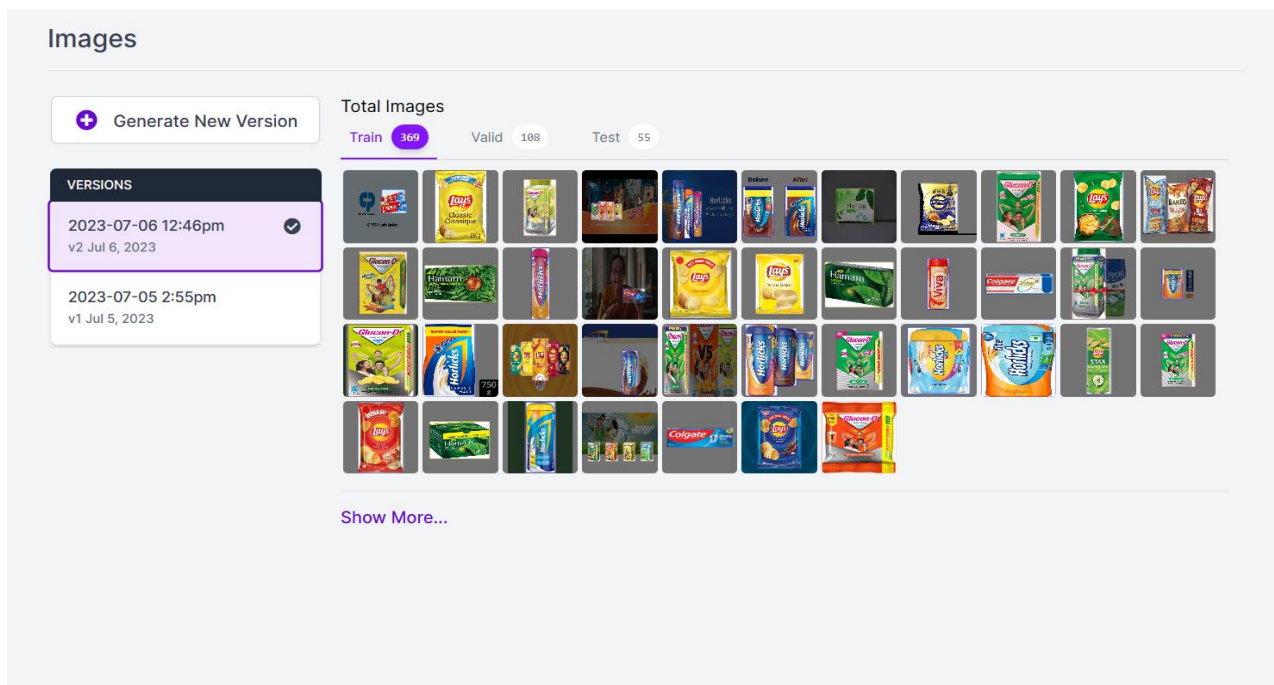


Fig. 10 Roboflow webpage

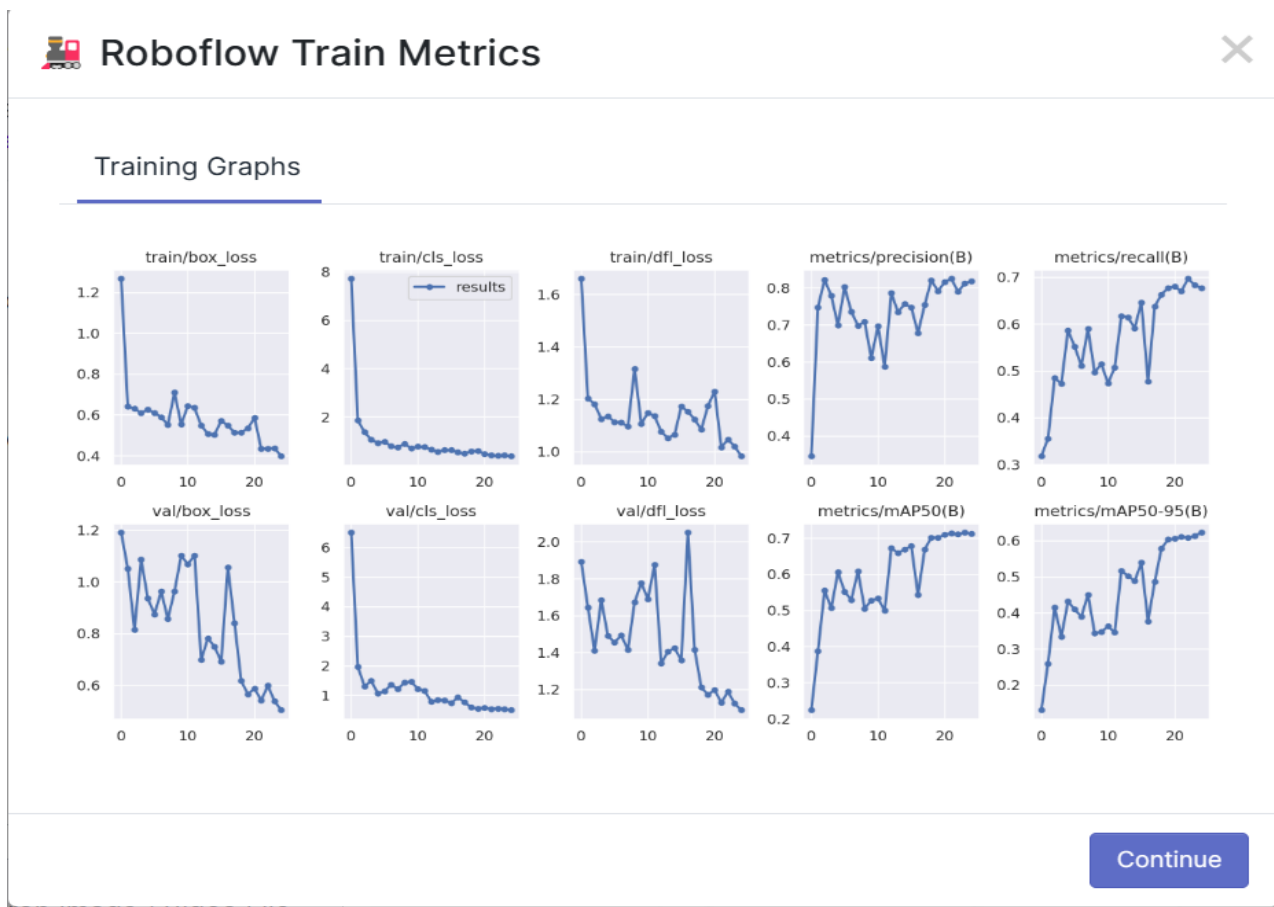


Fig. 11 Training graphs

Subsequently, a prediction script was developed to assess the model's performance in predicting objects within random images captured from the arena (see Fig. 12).



Fig.12 Sample inference

In parallel, I also read about the Robot Operating System (ROS) for the operation of Shobot. This helped me in writing a ROS publisher to access the camera attached to Shobot and retrieve individual frames. The code was further extended to implement object detection within each frame, enclosing identified objects in bounding boxes, and delivering the final output (see Fig. 13).

```

1  # Do not skip line 2
2  #!/usr/bin/env python
3
4  import rospy
5  import json
6  import cv2
7  from std_msgs.msg import String
8  #loading the model from a website using API key
9  from roboflow import Roboflow
10 rf = Roboflow(api_key="OGCV1pPJdtCRgU2K1s3r")
11 project = rf.workspace().project("jg-intern")
12 model = project.version(2).model
13
14
15
16 def publisher():
17     pub = rospy.Publisher('Parameters',String, queue_size=10)
18     rospy.init_node('Object_detector_model' anonymous=True)
19     rate = rospy.Rate(10)
20     #Starting the camera
21     cap = cv2.VideoCapture(0)
22     try:
23         while not rospy.is_shutdown():
24             ret, frame = cap.read()
25             #ret==1 means the frame is captured without any issue
26             if ret:
27                 #converting the image from BGR to RGB
28                 frame = cv2.cvtColor(src=frame, code=cv2.COLOR_BGR2RGB)
29                 #converting the prediction from a dictionary to string
30                 data = json.dumps(model.predict(frame,confidence=40, overlap=30).json())
31                 rospy.loginfo(data)
32                 pub.publish(data)
33                 rate.sleep()
34     finally:
35         #releasing the camera
36         cap.release()
37
38
39 if __name__ == '__main__':
40     try:
41         publisher()
42     except rospy.ROSInterruptException:
43         pass

```

Fig. 13 Data publishing script

## Revamped model and assisting with installation of the new SCARA arm

I augmented the dataset with an additional 500 images. Subsequent training was conducted to make use of the expanded dataset, refining the model's ability to predict and identify objects effectively.

Precision, an indicator of the model's accuracy in correctly identifying positive instances, improved from 81.8% to an impressive 94.9%. The mean averaged precision, a combined measure of the model's performance across all classes, showed a substantial increase, reaching a very good 90.9%. The model was tested with objects from arena and satisfactory results were obtained.

## JG intern Detection API

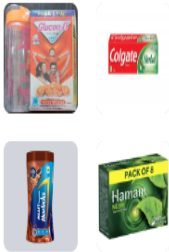
Use this pre-trained JG intern computer vision model to get predictions with our hosted API or deploy to the edge. [Learn More About Roboflow Inference](#)

2023-07-14 3:35pm Version 4 Generated Jul 14, 2023  
yolov8s Model Upload

yg-intern/4

90.9%	94.9%	85.8%
mAP	Precision	Recall

### SAMPLES FROM TEST SET



Visualize »

Drop Image / Video File

OR

Paste YouTube / Image URL

Try this model on images, video, or use your webcam

Confidence Threshold: 50%

0% 100%

Overlap Threshold: 50%

0% 100%

Label Display Mode:

Draw Confidence

Fig. 14 Model homepage with metrics





Fig. 15 Inference during training and testing

The dataset and the models were compressed into a single zip file thereby reducing space and easily transferable.

We tested the new SCARA arm with my system and learned how to make it move using different commands. We tried out different things with the arm, and it worked well. We checked its weight capacity, how far it could rotate, and how high it could lift. Everything

went smoothly, showing that the SCARA arm works effectively with my system. These tests helped us understand the arm better and figure out how to use it optimally in our setup.

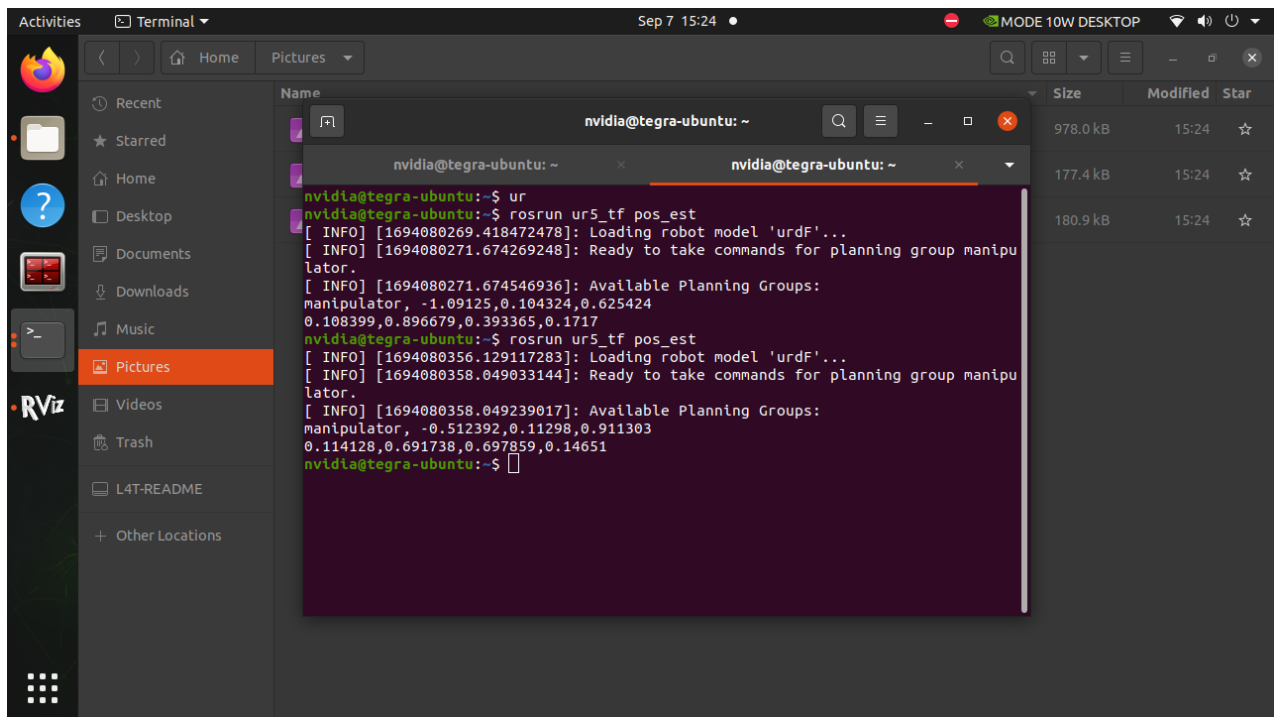


Fig. 16 New SCARA arm

## **BRAINSTORMING, EXPERIMENTATION AND THE RESULTS**

I explored the concept of robot coordinates and discovered different methods to manipulate them. One approach involved using a library named 'projectpixelto3darray,' which provides pixel coordinates with respect to the camera center. However, a drawback is that you need the specifications of the camera you are using. Another method I found involves manually calculating the frame length of the camera, which, although a bit tedious, is expected to yield accurate results.

I collaborated with Sai Teja to assist in a demo with the Accenture team. We found a camera link in the existing code but faced challenges in its utilization. After debugging together, we successfully obtained the runtime coordinates of the camera link. During manual testing on the Shobot, we found through experiment, the coordinates of the gripper and the camera at the home position. Despite initially interpreting 7 values as x, y, z, and angles, subsequent testing revealed that the first 3 coordinates did not form a right-handed system. Further exploration led us to understand that the 4 values were part of a quaternion, prompting the need for additional study.



```
nvidia@tegra-ubuntu: ~  
nvidia@tegra-ubuntu:~$ ur  
nvidia@tegra-ubuntu:~$ roslaunch ur5_tf pos_est  
[ INFO] [1694080269.418472478]: Loading robot model 'urdf'...  
[ INFO] [1694080271.674269248]: Ready to take commands for planning group manipulator.  
[ INFO] [1694080271.674546936]: Available Planning Groups:  
manipulator, -1.09125,0.104324,0.625424  
0.108399,0.896679,0.393365,0.1717  
nvidia@tegra-ubuntu:~$ roslaunch ur5_tf pos_est  
[ INFO] [1694080356.129117283]: Loading robot model 'urdf'...  
[ INFO] [1694080358.049033144]: Ready to take commands for planning group manipulator.  
[ INFO] [1694080358.049239017]: Available Planning Groups:  
manipulator, -0.512392,0.11298,0.911303  
0.114128,0.691738,0.697859,0.14651  
nvidia@tegra-ubuntu:~$
```

Fig. 17 Gripper coordinates

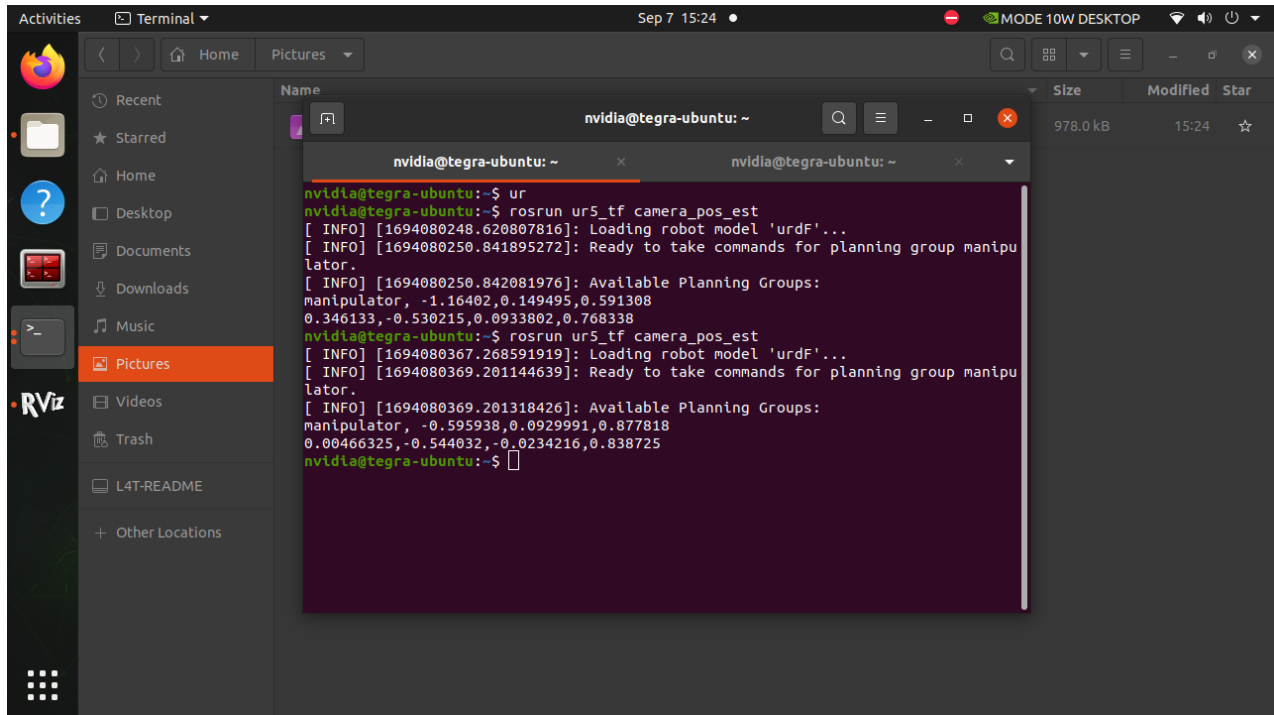


Fig. 18 Camera coordinates

In the final results, vertical and horizontal conversion factors were established, helpful in the measurements related to our Intel RealSense camera. We realized that the camera's frame of view resembles a rectangular pyramid. Keeping the distance between the camera and object to be 40 cm, we established the following frame size:

X-axis : 640 pixels, 43.5 cm  
Y axis : 480 pixels, 32.775 cm  
Z = 40 cm

The camera works as a rectangular pyramid, hence for any other distance parameters can be obtained by simply taking the ratio.



## Visual Results

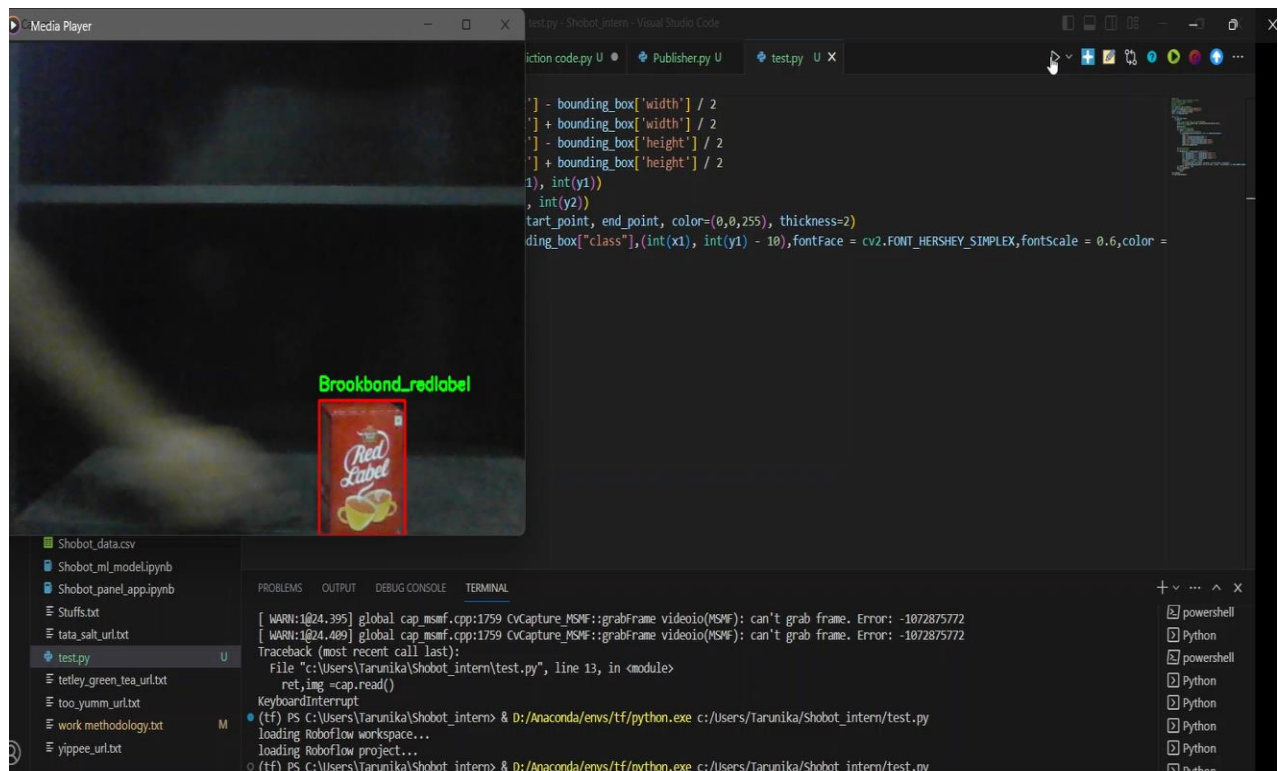


Fig. 19 Single object detection\_1

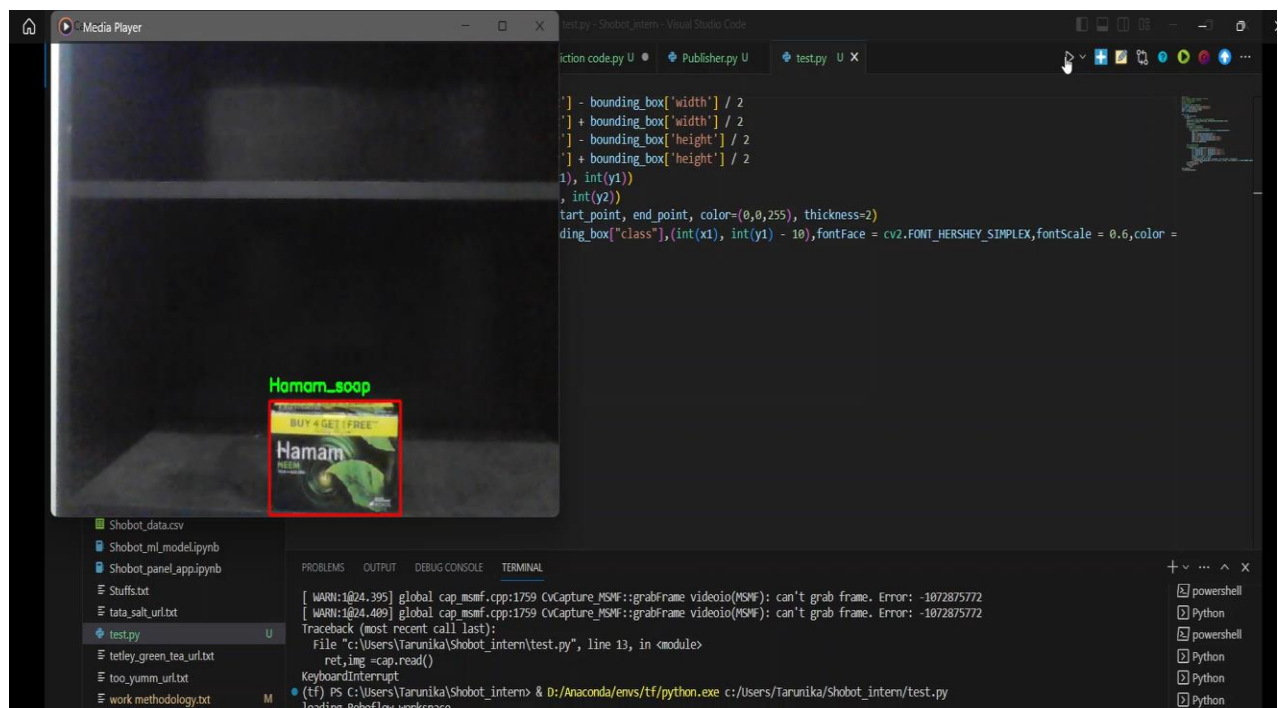


Fig. 20 Single object detection\_2

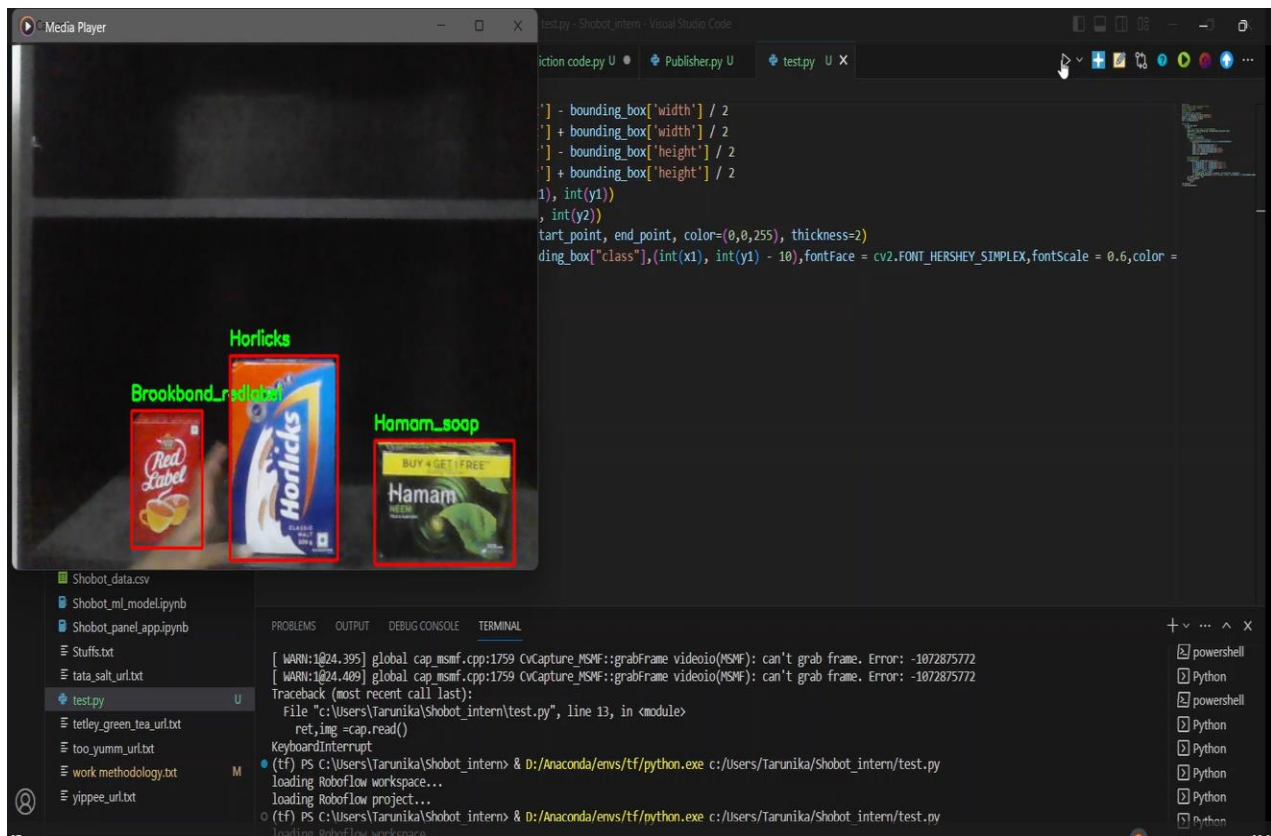


Fig. 21 Multiple objects detection

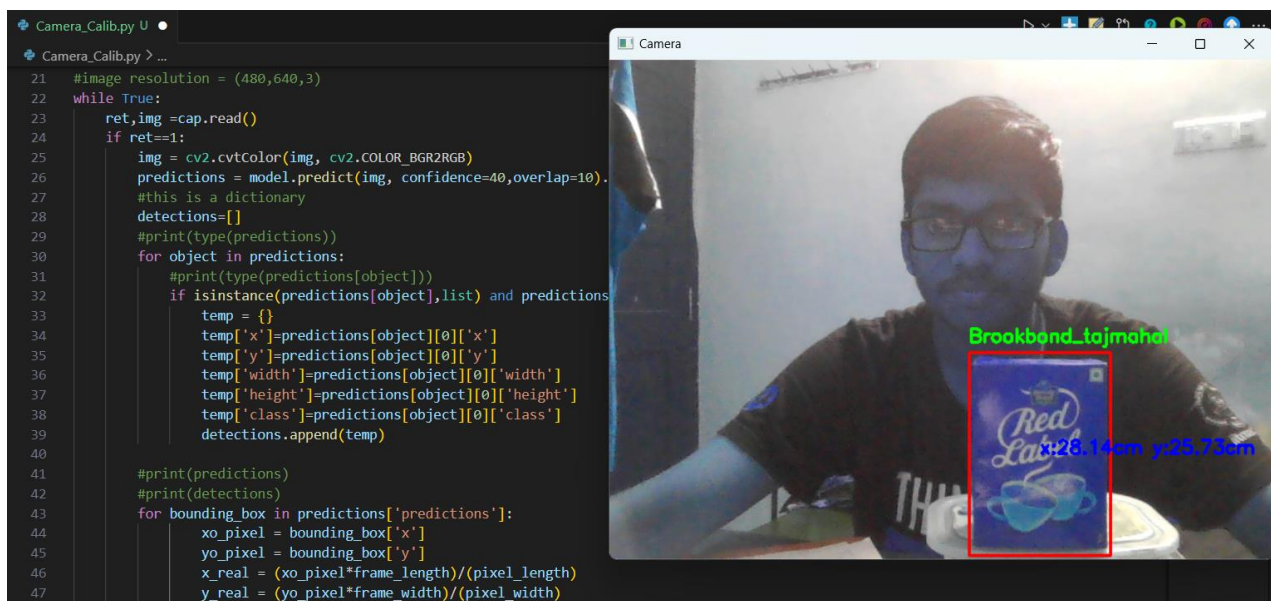


Fig. 22 Detection with location

## Conclusion

During my internship, I had the privilege of working under the guidance of Gokul Vishnu, who offered me invaluable insights, constructive feedback, and mentorship. Gokul's expertise and support played a pivotal role in my professional development and enhanced my understanding of the research process.

In summary, my time at CNDE has been incredibly fulfilling. It has strengthened my passion for research and my dedication towards machine learning and AI. The skills and knowledge I gained during this internship will undoubtedly form a solid foundation for my future academic and professional pursuits.

I extend my sincere gratitude to the entire CNDE team for granting me this valuable opportunity and for their constant support throughout my internship. I genuinely appreciate the enriching learning experience and the skills I've acquired during this period.

## Link to files

GitHub repository to files: [https://github.com/JG-0212/Shobot\\_Jayagowtham.git](https://github.com/JG-0212/Shobot_Jayagowtham.git)

## Literature and Reference

- [/https://pyimagesearch.com/](https://pyimagesearch.com/) , [GeeksforGeeks | A computer science portal for geeks](#), <https://docs.ultralytics.com/> were among my go-to websites during my internship