



# HI NUTS!

## User Guide

### ABSTRACT

This guide documents various steps to installation and usage of Hi Nuts application

Team

Zhou Zhibiao (A0269368J)

Sureshkumar Sujatha (A0269371W)

Tan Jun Ming (A0269372U)

Lim Zhi Jing (A0269638J)

Pattern Recognition System (Part- Time).

July 2023

## Summary

This guide provides steps for installation and usage of the app. Mac operating system is used for the preparation of the guide. Other operating system uses the same steps unless otherwise stated.

## Pre-requisites

# Operating System

1. Linux
2. Windows
3. Mac

# Required Programs

1. Anaconda (we have tested our application based on conda installation of python. Any standalone installation or other type of installation may require self troubleshooting if found any errors)
2. Python

## Install python libraries

Please use admin account for this application before installation and usage of the application.  
Example:

1. For windows, run the terminal as administrator
2. For linux and mac: use sudo su

```
sudo su
```

Please run following steps in the terminal

```
pip install Flask
pip install glob
pip install torch
pip install torchvision
pip install shutils
pip install Werkzeug
pip install pandas
pip install opencv-python
pip install numpy
pip install matplotlib
pip install ultralytics
pip install pycopy-xml.etree.ElementTree
pip install Pillow
pip install pathlib
pip install numpy --upgrade
```

## Hi Nuts App

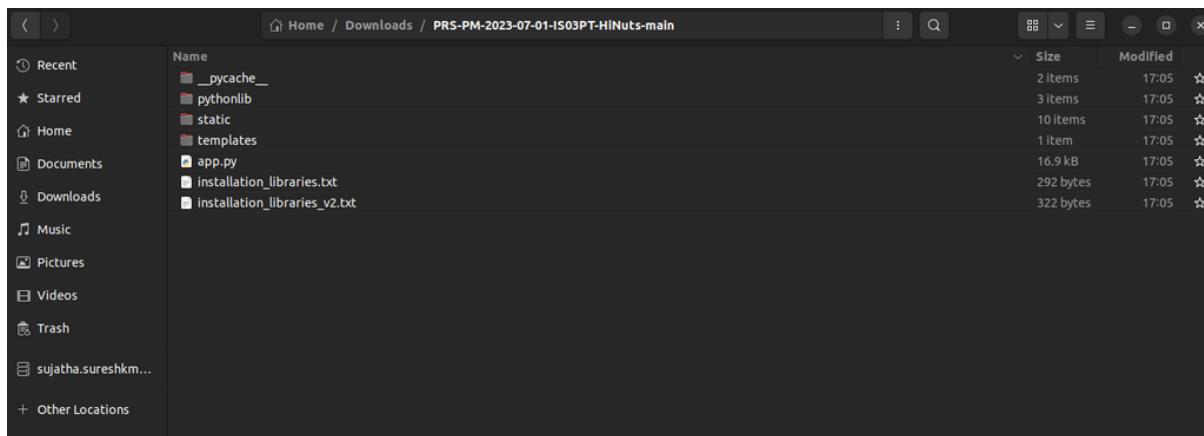
### Download App

Click [here](#) to download the Hi Nuts App source.

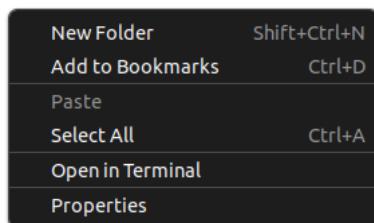
Download the source from the above mentioned source code link. Unzip the file “PRS-PM-2023-07-01-IS03PT-HiNuts-main.zip”

### Start App Hi Nuts

1. From Finder(Mac), Files(Linux), File Explorer (Windows): Go into the unzipped folder. Folder contents looks below



2. Right click and choose “Open in Terminal”



2. Please use admin account for this application before installation and usage of the application. Example:
  1. For windows, run the terminal as administrator
  2. For linux and mac: use sudo su
3. For Mac/Linux systems run the following commands:

```
export FLASK_APP=app
export FLASK_DEBUG=1
flask run
```

5. For Windows run the following commands:

```
set FLASK_APP=app
```

```
set FLASK_DEBUG=1
flask run
```

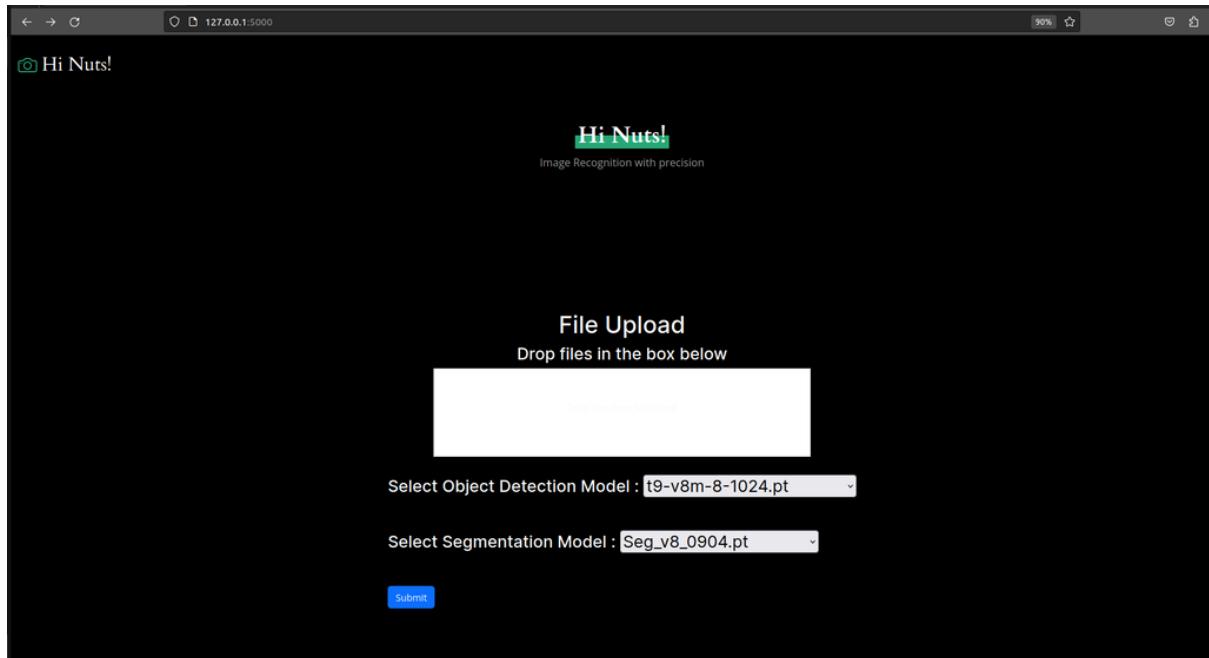
## 6. Identify the URL that is set by flask:

By default: <http://127.0.0.1:5000/>

```
sh-3.2$ flask run
[...]
[werkzeug] * Running on http://127.0.0.1:5000
[werkzeug] Press CTRL+C to quit
[...]
127.0.0.1 - - [16/May/2023 23:52:30] "GET / HTTP/1.1" 302 -
127.0.0.1 - - [16/May/2023 23:52:30] "GET /login HTTP/1.1" 200 -
127.0.0.1 - - [16/May/2023 23:52:30] "GET /assets/vendor/aos/aos.css HTTP/1.1" 404 -
127.0.0.1 - - [16/May/2023 23:52:30] "GET /assets/vendor/bootstrap/css/bootstrap.min.css HTTP/1.1" 404 -
127.0.0.1 - - [16/May/2023 23:52:30] "GET /static/css/bootstrap.min.css HTTP/1.1" 304 -
127.0.0.1 - - [16/May/2023 23:52:30] "GET /static/css/bootstrap.min.css HTTP/1.1" 304 -
127.0.0.1 - - [16/May/2023 23:52:30] "GET /assets/vendor/bootstrap-icons/bootstrap-icons.css HTTP/1.1" 404 -
127.0.0.1 - - [16/May/2023 23:52:30] "GET /assets/vendor/boxicons/css/boxicons.min.css HTTP/1.1" 404 -
127.0.0.1 - - [16/May/2023 23:52:30] "GET /assets/vendor/lightbox/css/lightbox.min.css HTTP/1.1" 404 -
127.0.0.1 - - [16/May/2023 23:52:30] "GET /assets/vendor/swiper/swiper-bundle.min.css HTTP/1.1" 404 -
127.0.0.1 - - [16/May/2023 23:52:30] "GET /assets/css/style.css HTTP/1.1" 404 -
```

## 7. Go to any web browser (Chrome/Safari/Firefox) and paste/type the URL given in flask. E.g.

Eureka!! Now you can access the Hi Nuts app.



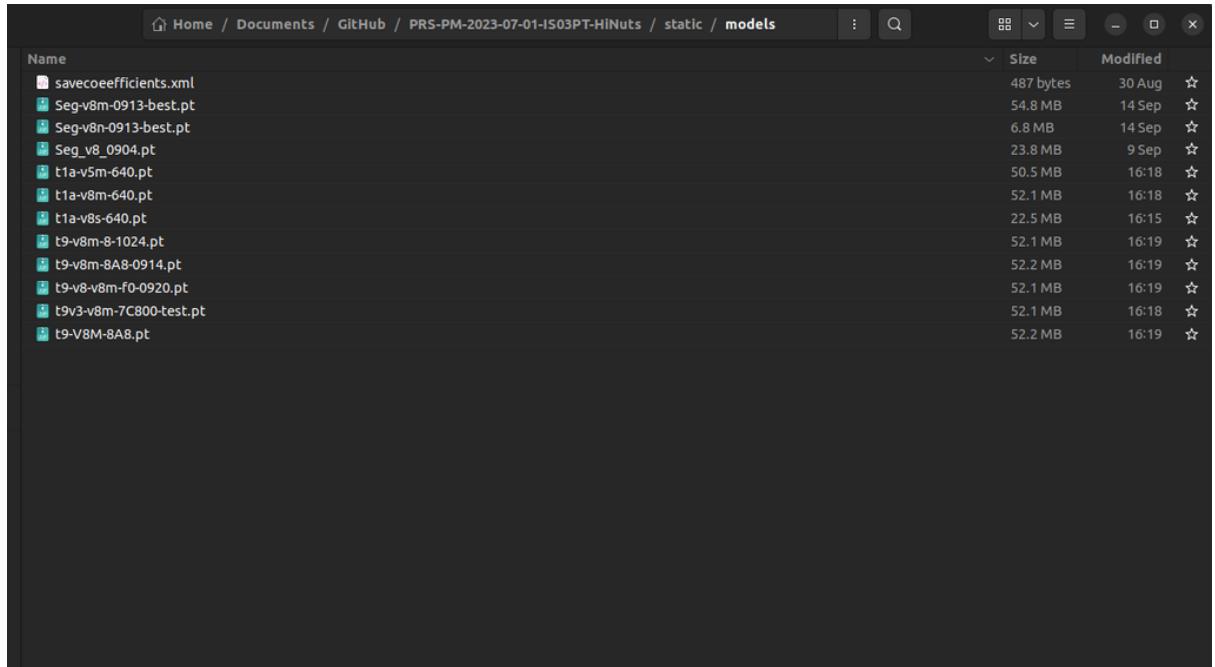
## Features

### Introduction Folder structure

1. Model files are stored in this folder location (**static/models**).

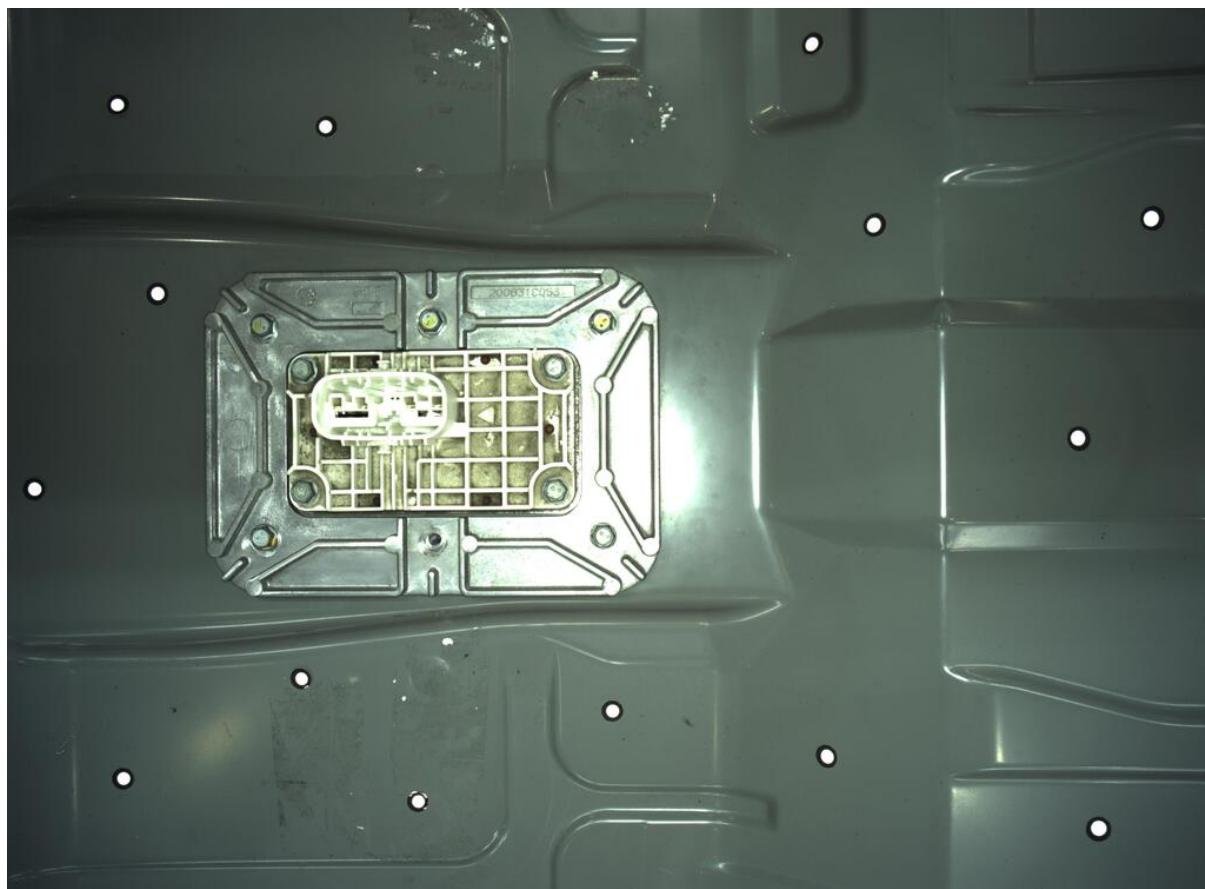
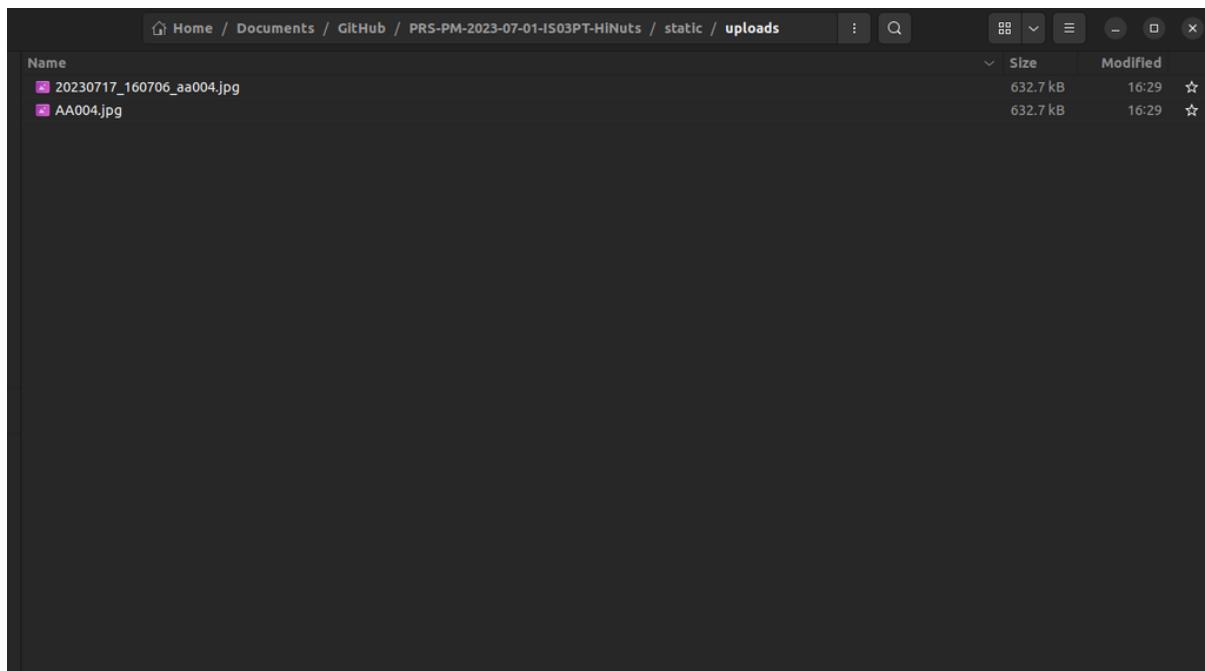
- **Usage:** model files will be provided by the data scientists shall be stored in this folder location.

Model file type	Format	Remarks
Camera Calibration	XML	File name should be “savecooefficients.xml“
Object Detection	pt	File name should start with “t”
Segmentation and Edge detection	pt	File name should start with “Seg“



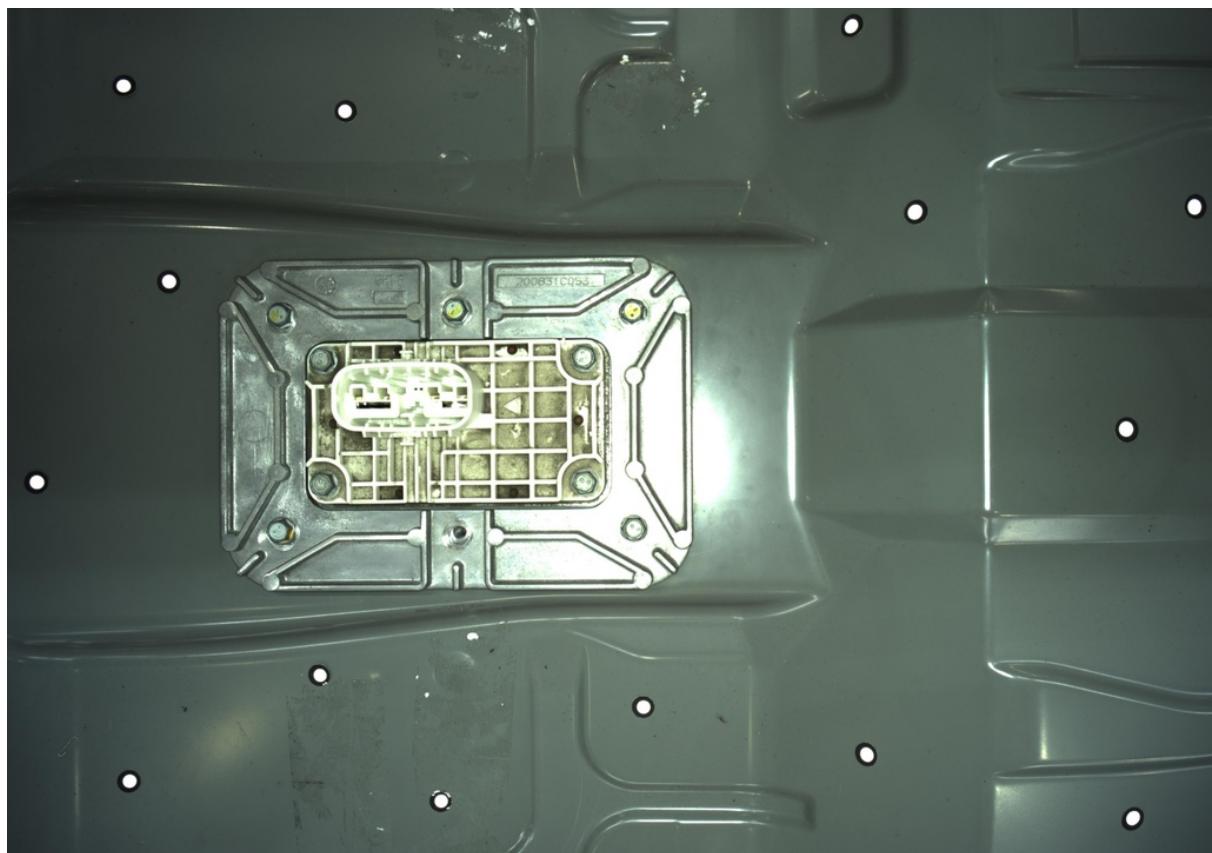
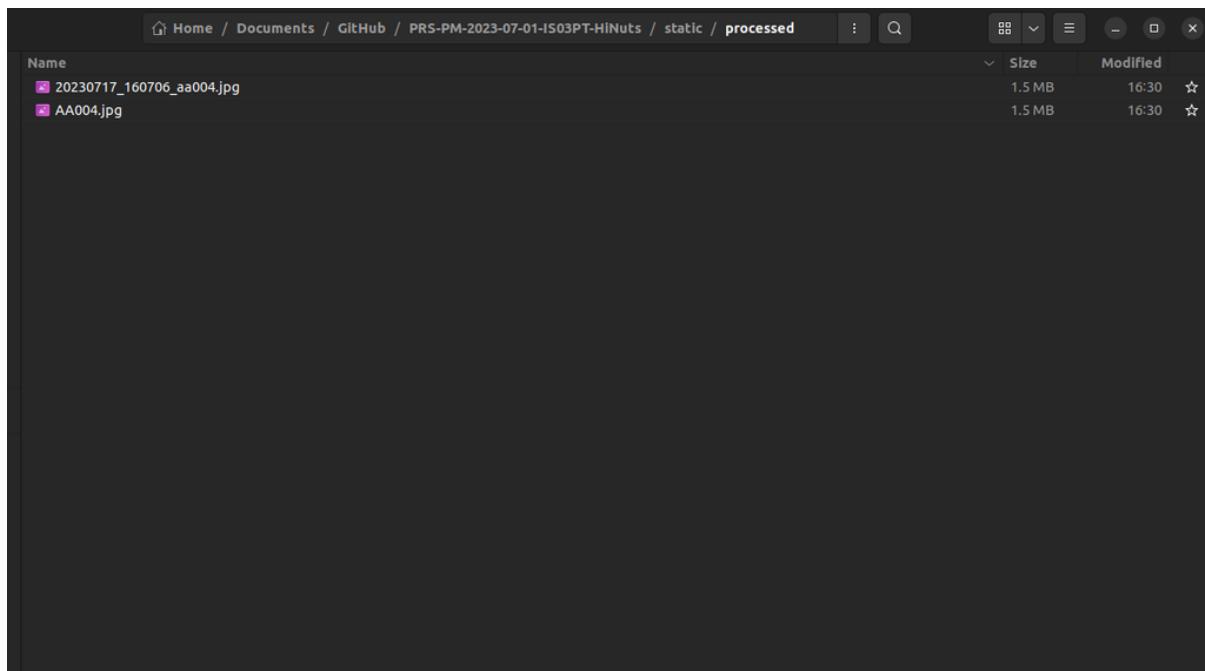
## 2. Upload Images

- **Usage:** When a user uploads a images it is stored in this location.
- **Folder location :** (static/uploads).



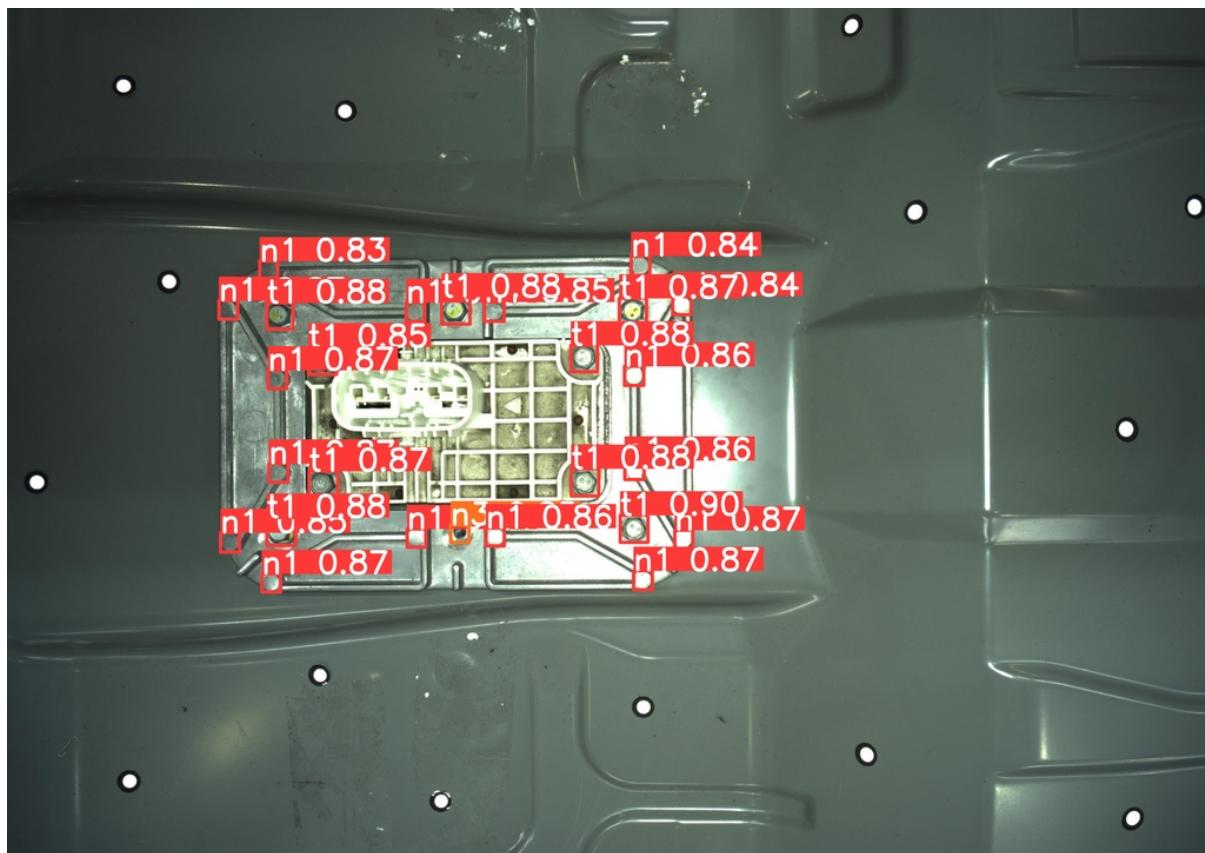
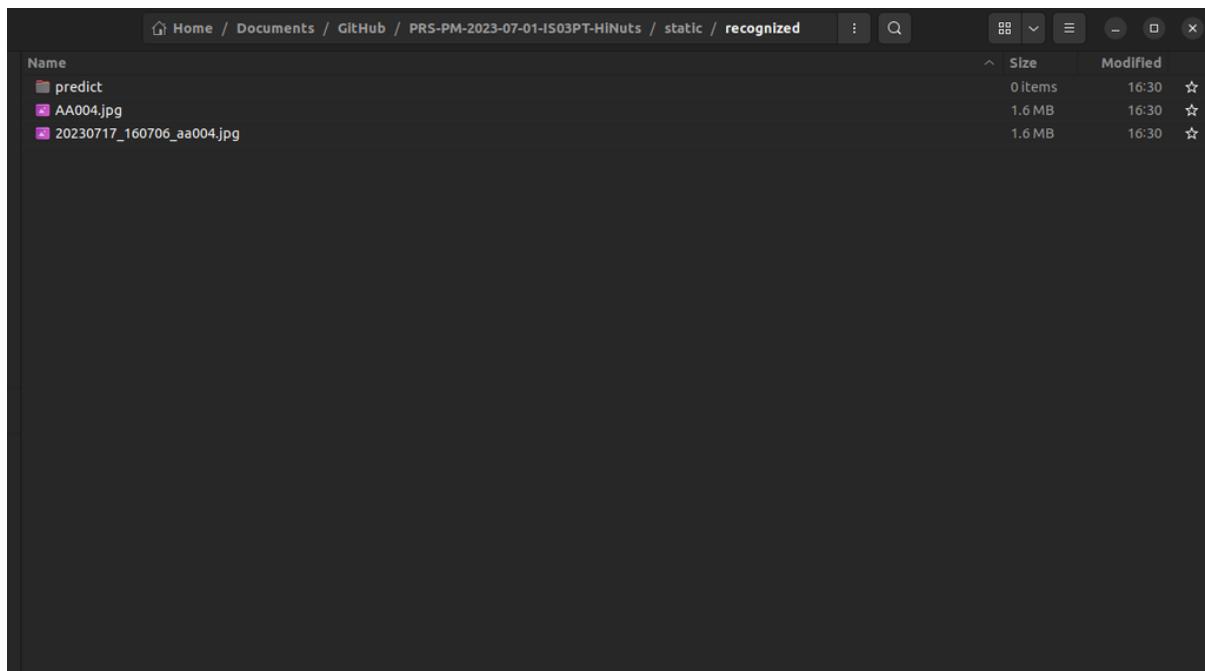
### 3. Processed Images

- **Usage:** Images are stored here after camera calibration
- **Folder location :** (static/processed).



#### 4. Recognized Images

- **Usage:** Images are stored here after object detection
- **Folder location :** (static/recognized).



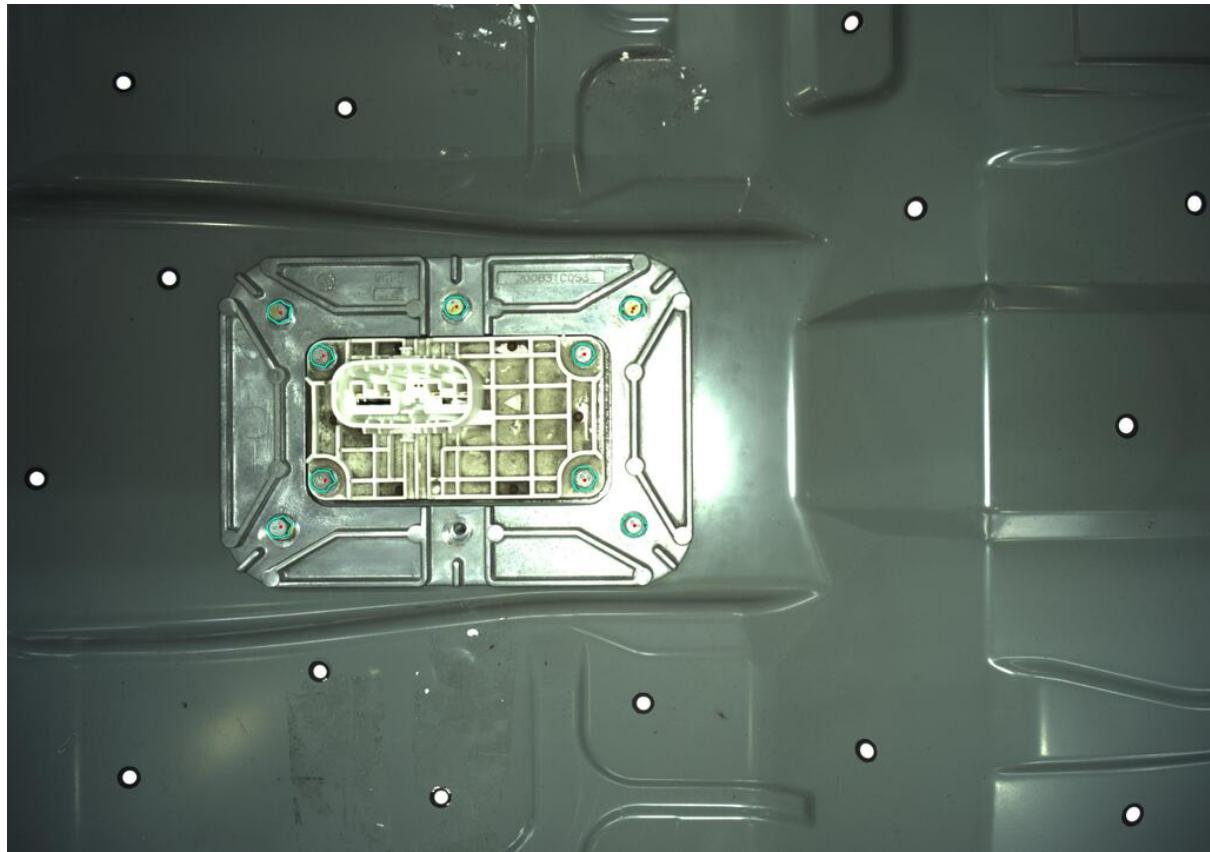
## 5. Segmentation Images

**Usage:** Images are stored here after segmentation and edge detection. A datapoints.csv is the final

output with the centre coordinates that is being shown in the UI are stored here.

## Folder location : (static/segment).

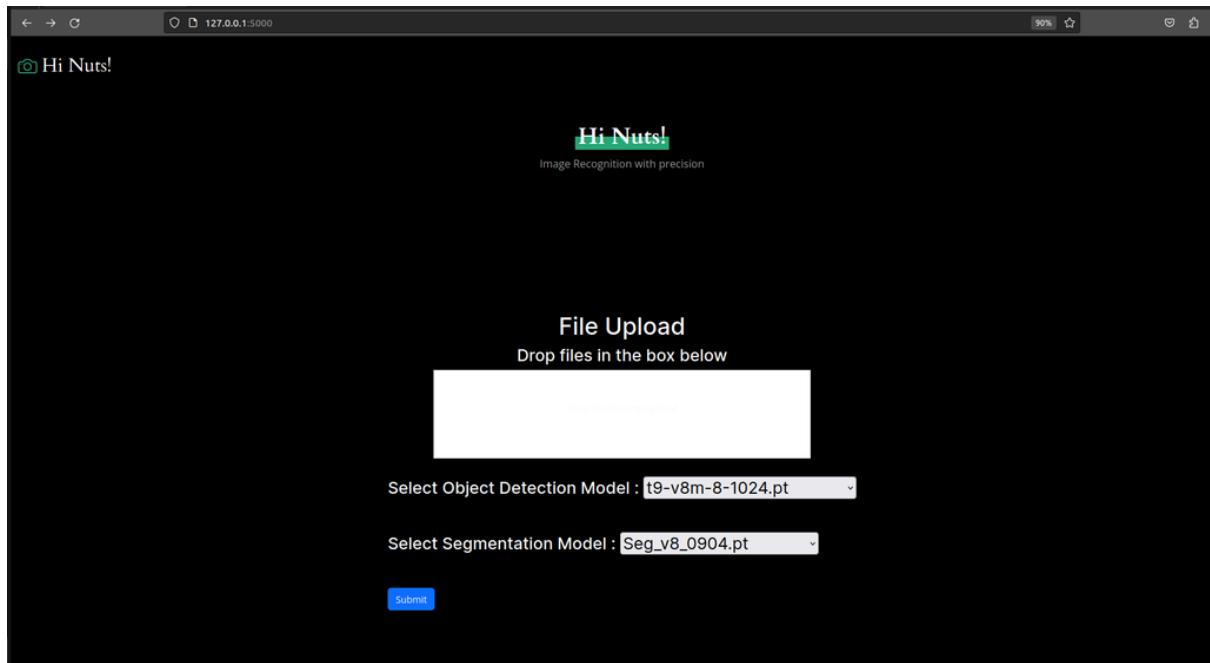
Name	Size	Modified
20230717_160706_aa004.jpg	582.2 kB	16:30
AA004.jpg	582.2 kB	16:30
datapoints.csv	3.1 kB	16:30



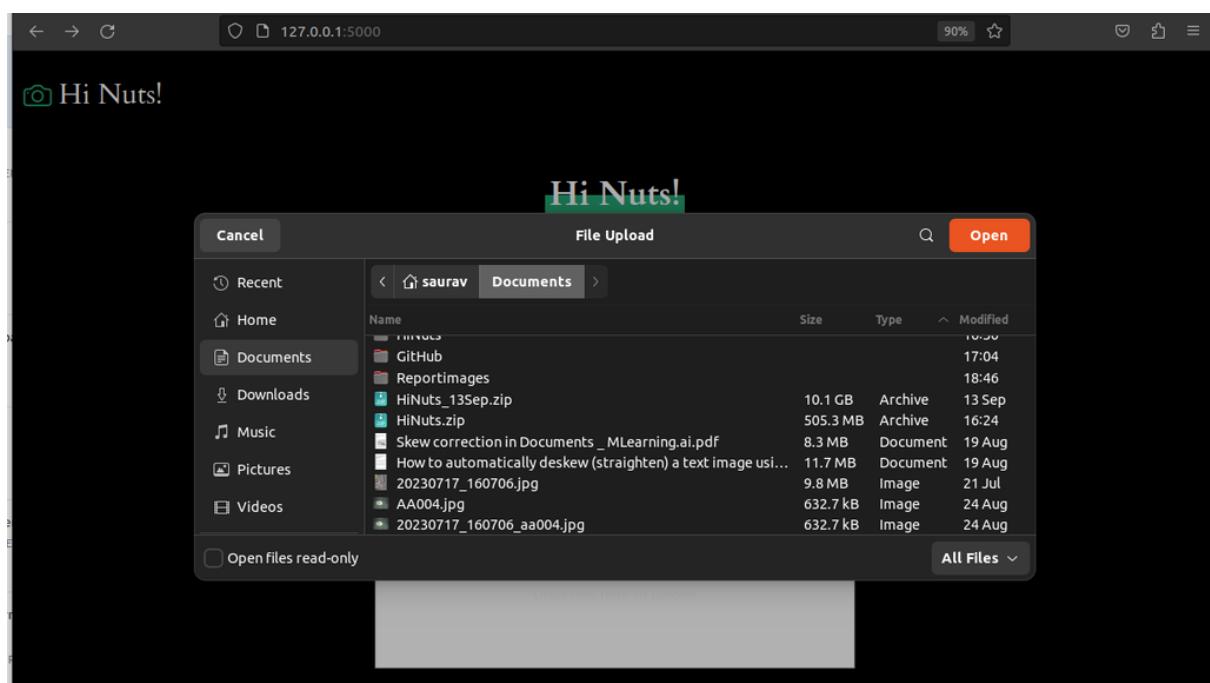
datapoints.csv		-/Documents/GitHub/PRS-PM-2023-07-01-1503PT-HINuts/static/segment
1	date_time,filename,x1,y1,x2,y2,confidence,class,x_final_centre_point,y_final_centre_point,zone	
2	2023-10-04 20:23:32,AA004.jpg,2614,1466064453125,1667,2110595703125,2105,01171875,1756,185546875,0,8990687131881714,0,0,2861,1465230871837,1713,58072622293499,0	
3	2023-10-04 20:23:32,AA004.jpg,1427,1319580678125,951,16162169375,1526,39892578125,1646,2347412169375,0,884287416934967,0,0,1476,6552371242789,994,4757149843562,1	
4	2023-10-04 20:23:32,AA004.jpg,853,3138427734375,971,2120361328125,930,9459228515625,1656,767944359375,0,8810753226286212,0,0,897,1228461391681,1013,4533357746276,8	
5	2023-10-04 20:23:33,AA004.jpg,1853,3253173828125,1514,44287109375,1943,36181040625,1669,4473876953125,0,8798049688339233,0,0,1897,951426323989,1564,869782858456,0	
6	2023-10-04 20:23:33,AA004.jpg,851,9385375976562,1674,3087158203125,948,0213012695312,1761,7362060546875,0,8775622844696645,0,0,899,7852959862121,1717,9328659257772,7	
7	2023-10-04 20:23:33,AA004.jpg,1852,1961669921875,1169,28173828125,1939,6973876953125,1194,128173828125,0,877302056590515,0,0,1896,3043118219698,1150,2202210319176,7	
8	2023-10-04 20:23:33,AA004.jpg,2611,883056640625,955,69130859375,2106,16664453125,1039,9969482421875,0,8703851699829102,0,0,2057,9324967887815,1006,5287366579365,1	
9	2023-10-04 20:23:34,AA004.jpg,991,9769287109375,1522,45458984375,1084,6923828125,1612,859136859375,0,8695214986801147,0,0,1045,6376195373775,1564,924714544284,7	
10	2023-10-04 20:23:34,AA004.jpg,990,9647827148438,1115,058837890625,1082,9580078125,1205,4600830878125,0,8485365569986877,0,0,1039,6336524126664,1155,7680372112282,7	

# App Usage

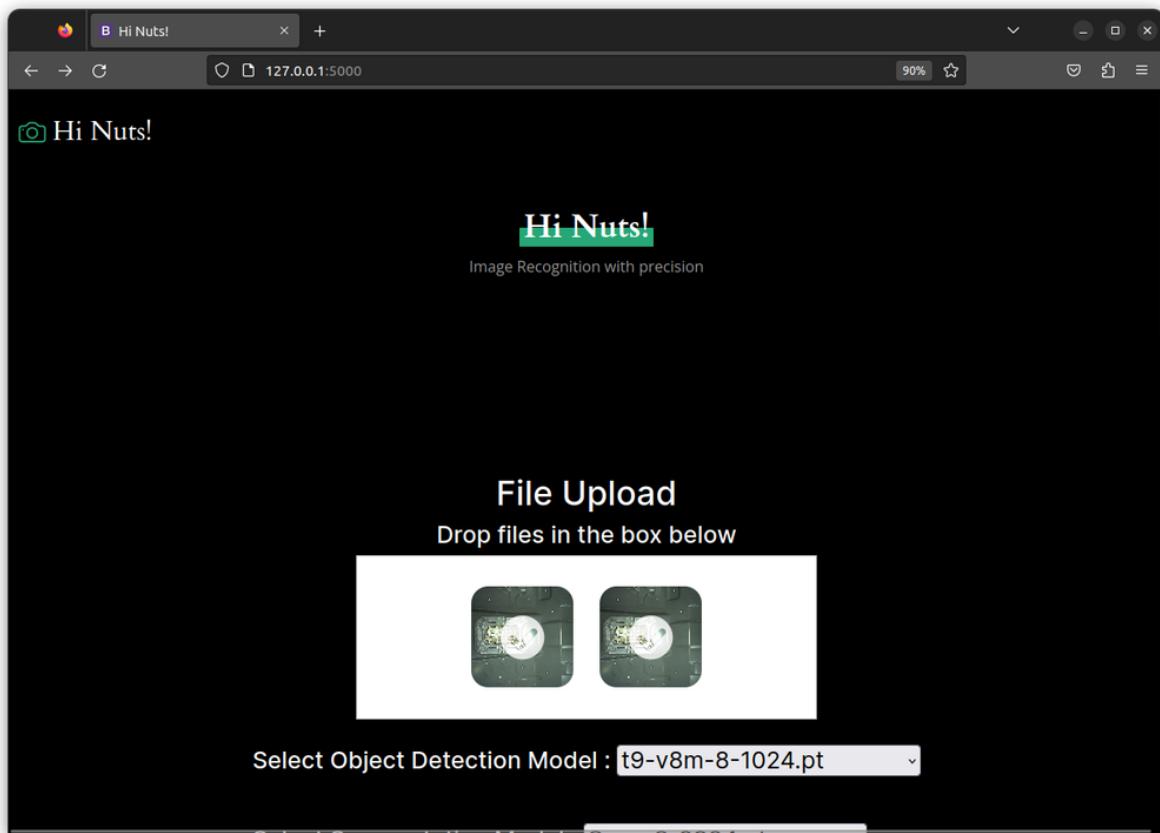
1. Go to the main page: By default: <http://127.0.0.1:5000/>



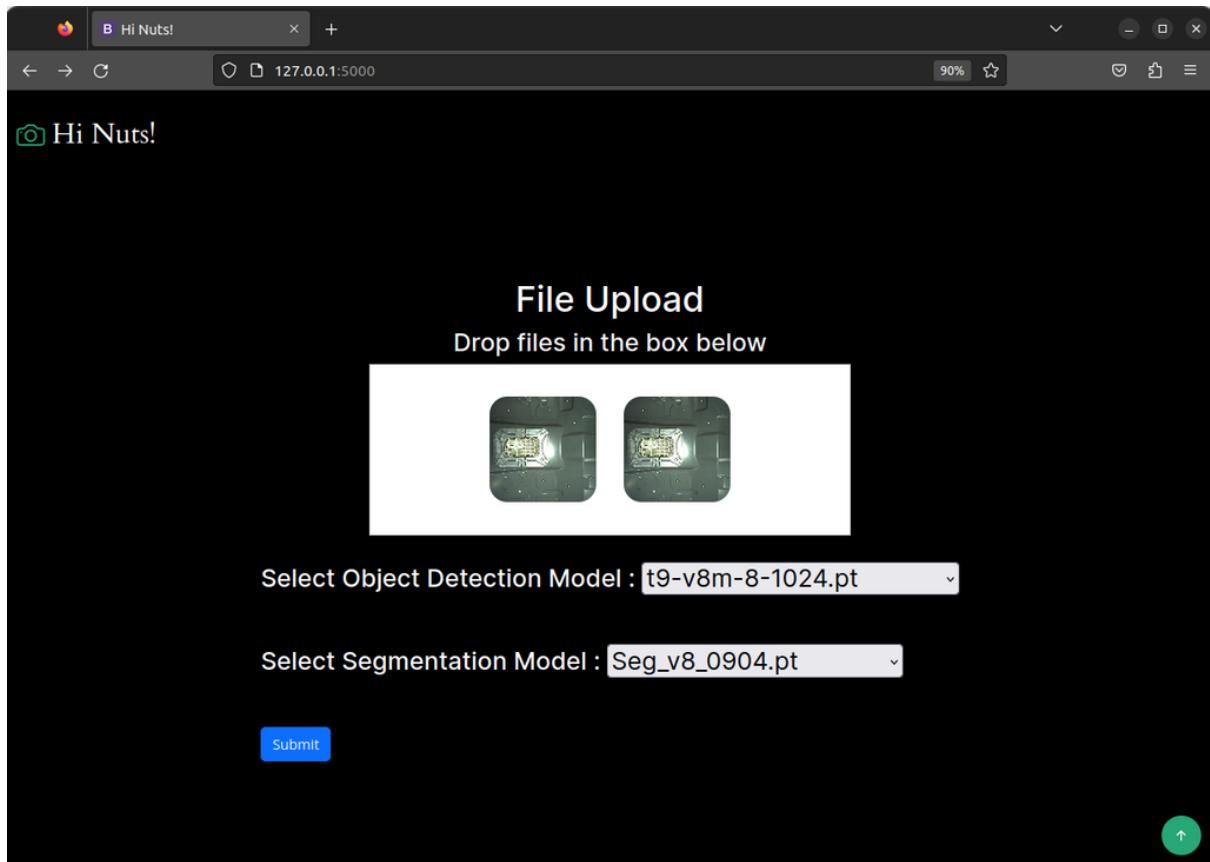
2. Click the white box and file explorer option will open up to choose the images.



3. When the images are uploaded sucessfully and the tick mark shown as the images are ready for processing.



4. Choose the object detection model and segmentation model that is available from the dropdown and click submit.



5. You will be able to monitor the process from the terminal. wait for the output "127.0.0.1 - - [04/Oct/2023 20:04:55] "POST / HTTP/1.1" 200 -". this shows the whole process is complete.

```
root@saurav-XPS-13-9360: /home/saurav/Documents/GitHub/PRS-PM-2023-07-01-IS03PT-HINuts
Center Point: (43.241299641815054)
1853.3253173828125 1514.44287109375 1943.36181640625 1004.4473876953125

0: 160x160 1 t1_seg, 46.9ms
Speed: 0.4ms preprocess, 46.9ms inference, 1.4ms postprocess per image at shape (1, 3, 160, 160)
Results saved to runs/segment/predict
Center Point: (44.62610294117647, 50.36691176470588)
853.9385375976562 1674.3087158203125 948.02130126953125 1766.7362060546875

0: 160x160 1 t1_seg, 48.5ms
Speed: 0.5ms preprocess, 48.5ms inference, 1.6ms postprocess per image at shape (1, 3, 160, 160)
Results saved to runs/segment/predict
Center Point: (45.84675838255876, 43.62415010546461)
1852.1961669921875 1109.28173828125 1939.6973876953125 1196.128173828125

0: 160x160 1 t2_seg, 47.6ms
Speed: 0.4ms preprocess, 47.6ms inference, 2.0ms postprocess per image at shape (1, 3, 160, 160)
Results saved to runs/segment/predict
Center Point: (44.108144829782205, 40.93848275066762)
2011.883056640625 955.09130859375 2100.16064453125 1039.9969482421875

0: 160x160 1 t1_seg, 47.7ms
Speed: 0.6ms preprocess, 47.7ms inference, 1.7ms postprocess per image at shape (1, 3, 160, 160)
Results saved to runs/segment/predict
Center Point: (46.68944014815645, 45.43742806418648)
992.9769287109375 1522.45458984375 1086.6923828125 1612.859130859375

0: 160x160 1 t1_seg, 50.8ms
Speed: 0.4ms preprocess, 50.8ms inference, 2.5ms postprocess per image at shape (1, 3, 160, 160)
Results saved to runs/segment/predict
Center Point: (52.6666982644081, 42.47012470053427)
998.9647827148438 1115.05883789625 1082.9580078125 1205.4600830078125

0: 160x160 1 t2_seg, 45.5ms
Speed: 0.4ms preprocess, 45.5ms inference, 1.8ms postprocess per image at shape (1, 3, 160, 160)
Results saved to runs/segment/predict
Center Point: (48.668869697822714, 40.70919932059522)
1427.1319580078125 951.16162109375
2011.883056640625 955.09130859375
df2.shape PRE (9, 11)
df.shape (9, 17)
df2.shape POST (18, 11)
static/recognized/predict/A0004.jpg
static/recognized/predict/20230717_160706_aa004.jpg
127.0.0.1 - - [04/Oct/2023 20:04:55] "POST / HTTP/1.1" 200 -
```

6. UI automatically refreshes to show the output. **You may need to scroll down for the output.**

Output Type	Output Definition	Remarks
Original	This is the image that is uploaded by the user	
Prediction	This is the image after image calibration and object detection	Output after Model 1 & 2
Segmentation	This is the image after Segmentation and image detection	Output after Model 3



6. Data Output in table format. The rows reflects number of objects detected. during the process of the detection there can be both relevant nuts and noises. we discard the noises and capture only relevant nuts.

Output Name	Output Description
Index	number of objects detected
date_time	date time when the processing is done
x1	detected object left x boundary
y1	detected object left y boundary
x2	detected object right x boundary
y2	detected object right y boundary
confidence	confidence of the object detection
class	class identified
x_final_centre_point	x coordinate centre point
y_final_centre_point	y coordinate centre point
zone	zone identified that the robot goes in sequence.

	date_time	filename	x1	y1	x2	y2	confidence	class	x_final_centre_point	y_final_centre_point	zone
0	2023-10-04 20:11:12	AA004.jpg	2014.146606	1667.211060	2105.011719	1756.185547	0.899069	0.0	2061.140523	1713.500726	0
1	2023-10-04 20:11:12	AA004.jpg	1427.131958	951.161621	1520.398926	1040.234741	0.884287	0.0	1476.655237	994.475715	1
2	2023-10-04 20:11:13	AA004.jpg	853.313843	970.212036	939.945923	1056.767944	0.881075	0.0	897.122846	1013.453336	8
3	2023-10-04 20:11:13	AA004.jpg	1853.325317	1514.442871	1943.361816	1604.447388	0.879805	0.0	1897.951420	1564.809783	0
4	2023-10-04 20:11:13	AA004.jpg	853.938538	1674.308716	948.021301	1766.736206	0.877562	0.0	899.785296	1717.932866	7
5	2023-10-04 20:11:13	AA004.jpg	1852.196167	1109.281738	1939.697388	1196.128174	0.877302	0.0	1896.304312	1150.220221	0
6	2023-10-04 20:11:13	AA004.jpg	2011.883057	955.091309	2100.160645	1039.996948	0.870385	0.0	2057.932497	1000.528737	1
7	2023-10-04 20:11:14	AA004.jpg	992.976929	1522.454590	1086.692383	1612.859131	0.869521	0.0	1045.637620	1564.924715	7
8	2023-10-04 20:11:14	AA004.jpg	990.964783	1115.058838	1082.958008	1205.460083	0.848537	0.0	1039.633652	1155.768037	7