

A Mini-Project Report
on

SMART CAMERA

Submitted for partial fulfilment of the requirements for the award of the degree
of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

BY

SHHREYYA SRI (2451-19-733-076)
VENKATESH GOUD(2451-19-733-077)

Under the guidance of

Dr. Sandhya Banda
<Designation>



Department of Computer Science and Engineering
Maturi Venkata Subba Rao Engineering College
(An Autonomous Institution)
(Affiliated to Osmania University & Recognized by AICTE)
Nadergul(V), Balapur(M), RR Dist. Hyderabad – 501 510

2021-22.

Maturi Venkata Subba Rao Engineering College **(An Autonomous Institution)**

(Affiliated to Osmania University, Hyderabad)
Nadargul(V), Hyderabad-501510



Certificate

This is to certify that the mini-project work entitled
"SMART CAMERA"

is a bonafide work carried out by **SHHREYYA SRI (2451-19-733-076)**
VENKATESH GOUD(2451-19-733-077)

in partial fulfillment of the requirements for the award of degree of **BACHELOR OF ENGINEERING IN COMPUTER SCIENCE AND ENGINEERING** from Maturi Venkata Subba Rao Engineering College, affiliated to OSMANIA UNIVERSITY, Hyderabad, under our guidance and supervision.

The results embodied in this report have not been submitted to any other university or institute for the award of any degree or diploma.

Internal Guide
Dr. B.Sandhya

Project Co-Ordinators
Dr. B.Sandhya, Professor
G.Vijay Kumar, Associate Professor

External Examiner

DECLARATION

This is to certify that the work reported in the present mini-project entitled **“smart camera”** is a record of bonafide work done by us/ me in the Department of Computer Science and Engineering, Maturi Venkata Subba Rao Engineering College, Osmania University. The reports are based on the mini-project work done entirely by us and not copied from any other source.

The results embodied in this mini-project report have not been submitted to any other University or Institute for the award of any degree or diploma to the best of our/ my knowledge and belief.

SHHREYYA SRI
2451-19-733-076

VENKATESH GOUD
2451-19-733-077

Table Of Contents

S No.		Pg No.
1.	INTRODUCTION	
	1.1 PROBLEM STATEMENT	5
	1.2 SCOPE OF THE SMART CAMERA	6
2.	TOOLS AND TECHNOLOGIES	
	2.1 HARDWARE REQUIRMENTS	7
	2.2 SOFTWARE REQUIRMENTS	8
3.	DESIGN	
	3.1 FLOW CHARTS	9
	3.2 AI MODEL	10
4.	IMPLEMETATION	
	4.1 ALGORITHM	11
	4.2 CODE	12
5.	TESTING / RESULTS	
	5.1 TEST CASES (SCREENSHOTS) / RESULTS (SCREENSHOTS)	18
6.	FUTURE SCOPE	23
7.	CONCLUSIONS	24

INTRODUCTION

1.1 PROBLEM STATEMENT

Smart camera detects a person who is in the camera vision.

When the person is detected, it starts recording and when the person goes out of the camera vision it stops recording and these recordings are stored in local database.

1.2 SCOPE OF THE PROJECT

- Smart camera enables the user to store the video which is necessary .
- Reduces the data to be stored.
- Helpful in place of CCTV and also for personal use

TOOLS AND TECHNOLOGIES

2.1 HARDWARE REQUIRED

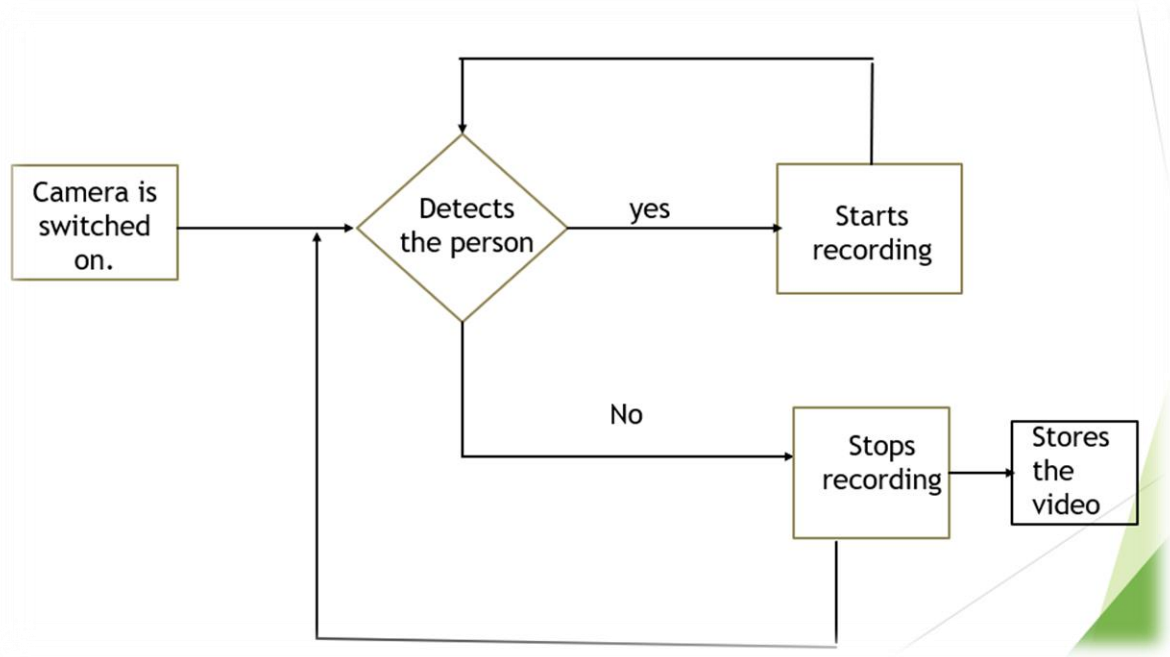
- Laptop with camera / Desktop with webcam.
- 2 GB RAM minimum.
- Storage internal / external

2.2 SOFTWARE REQUIRED

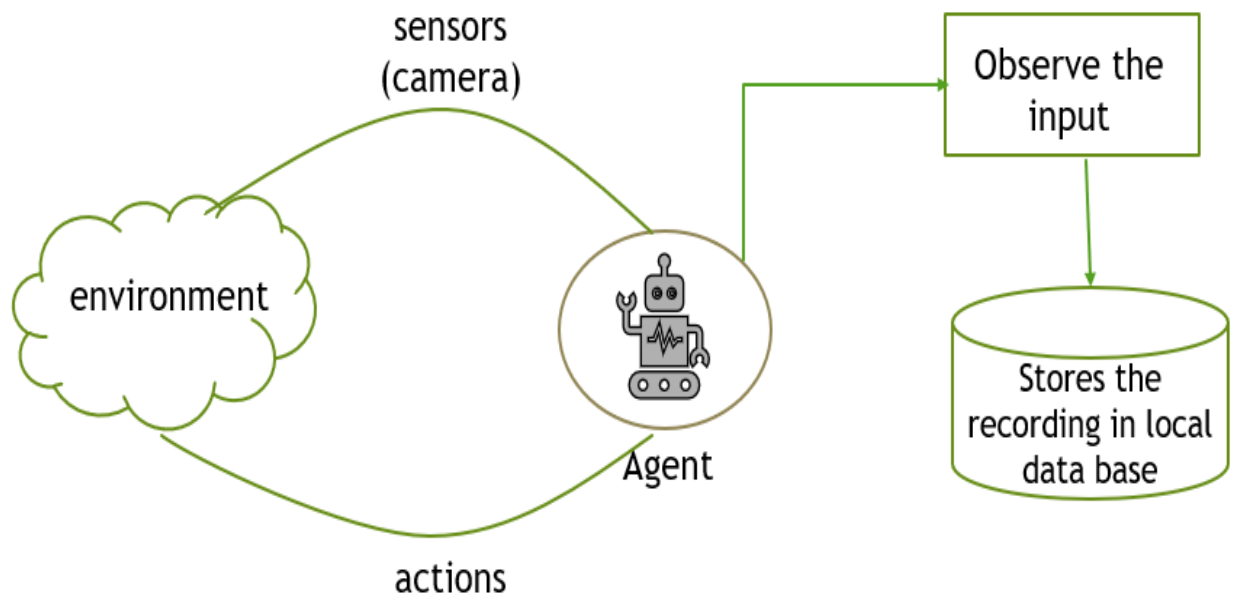
- Python toolkit.
- Visual studio IDE.
- OpenCV package.

DESIGN REVIEW

3.1 FLOW CHART



3.2 AI MODEL



IMPLEMENTATION

4.1 ALGORITHM

1. Set up the camera.
2. Include haarcascade facial and body XML files.
3. Take the frames of the input video and covert into gray scale.
4. If any body or face is detected in the frame
 - Then set detection as true
 - Start recording the video with file name as date and time.
 - Print start recording !
5. If detection is false wait for 5 seconds and stop the recording
 - Print Recording stopped !
6. Go to step 4 until 'q' is pressed.

4.2 CODE

```
import cv2
import time
import datetime
```

cv2 module is imported for image processing and performing computer vision tasks and time, datetime module provides a function for getting local time and date.

```
cap = cv2.VideoCapture(0)
```

we use videocapture() function to capture live stream with webcam. We need to create a object of videocapture() to capture a video.

```
face_cascade = cv2.CascadeClassifier(
    cv2.data.harcascades + "haarcascade_frontalface_default.xml")

body_cascade = cv2.CascadeClassifier(
    cv2.data.harcascades + "haarcascade_fullbody.xml")
```

CascadeClassifier is used to load a .xml classifier file. Frontalface and fullbody are used for face and body detection.

```
detection = False
detection_stopped_time = None
timer_started = False
SECONDS_AFTER_DETECTION = 5
```

Detection Variable is declared to check whether the face or body is detected or not

Detection_stopped_time is used to store the recording stopped time and timer_started is used whether recording started or not.

```
f_size = (int(cap.get(3)), int(cap.get(4)))
fourcc = cv2.VideoWriter_fourcc(*"mp4v")
```

f_size is the frame size. Cap.get(3) gives the frame width and cap.get(4) gives the frame height. The default resolutions of the frame are obtained, and these are system dependent, so we convert the resolutions from float to integer. FourCC is a 4-byte code used to specify the video codec. We have used mp4v codec for mp4 video.

```
while True:
    _, frame = cap.read()

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

We write rest of the code in while loop, for repetitive checking of frames and recording the video. We use read() function to read a frame from the video and store that in 'frame' variable. This frame is converted into gray scale. Grayscale simplifies the algorithm and reduces computational requirements.

read() function will return 2 objects, but we are interested only in the latest one, that is the last frame from the camera. So we are ignoring the first object _, and naming the second as frame, that we will use to feed our detect() function in a second.

```
faces = face_cascade.detectMultiScale(gray, 1.3, 5)
bodies = body_cascade.detectMultiScale(gray, 1.3, 5)
```

MultiScale detects objects of different sizes in the input image and returns rectangles positioned on the faces. The first argument is the image, the second is the scalefactor (how much the image size will be reduced at each image scale), and the third is the minNeighbors (how many neighbors each rectangle should have). The values of 1.3 and 5 are based on experimenting and choosing those that worked best.

```
if len(faces) + len(bodies) > 0:
    if detection:
        timer_started = False
    else:
        detection = True
        current_time = datetime.datetime.now().strftime("%d-
%m-%Y-%H-%M-%S")

    out = cv2.VideoWriter(
        f"{current_time}.mp4", fourcc, 20, f_size)
    print("Started Recording!")
```

In each frame we check either any body or face is detected. If frame contains face or body is detected, we set the detection variable as true and start saving the recording with the name with current date and time.

The current date and time is obtained using now() and in the mentioned format as date-month-year-hours-minutes-seconds.

VideoWriter() object will create a frame of the video input and store with the file name current_time. It take four inputs file name, fourcc, fps and frame size.

A print statement is there, which prints recording has been started on the console.

If the detection is already true, we make the time_started variable as false.

```
elif detection:
    if timer_started:
        if time.time() - detection_stopped_time
>=SECONDS_AFTER_DETECTION:
            detection = False
            timer_started = False
            out.release()
            print('Recording Stopped !')
    else:
        timer_started = True
        detection_stopped_time = time.time()
```

If there is no body or face detected and detection is true, we enter into ifelse loop. Where We check for the timer has been started or not. If timer_started is true, we check with present time. If the time is more than 5 seconds we make detection and timer_started as false and stop the recording.

We print a statement that recording has been stopped.

If there is no body or face detected, detection is true and timer_started is false, then we make timer_started as true and note down the detection stopped time in a variable.

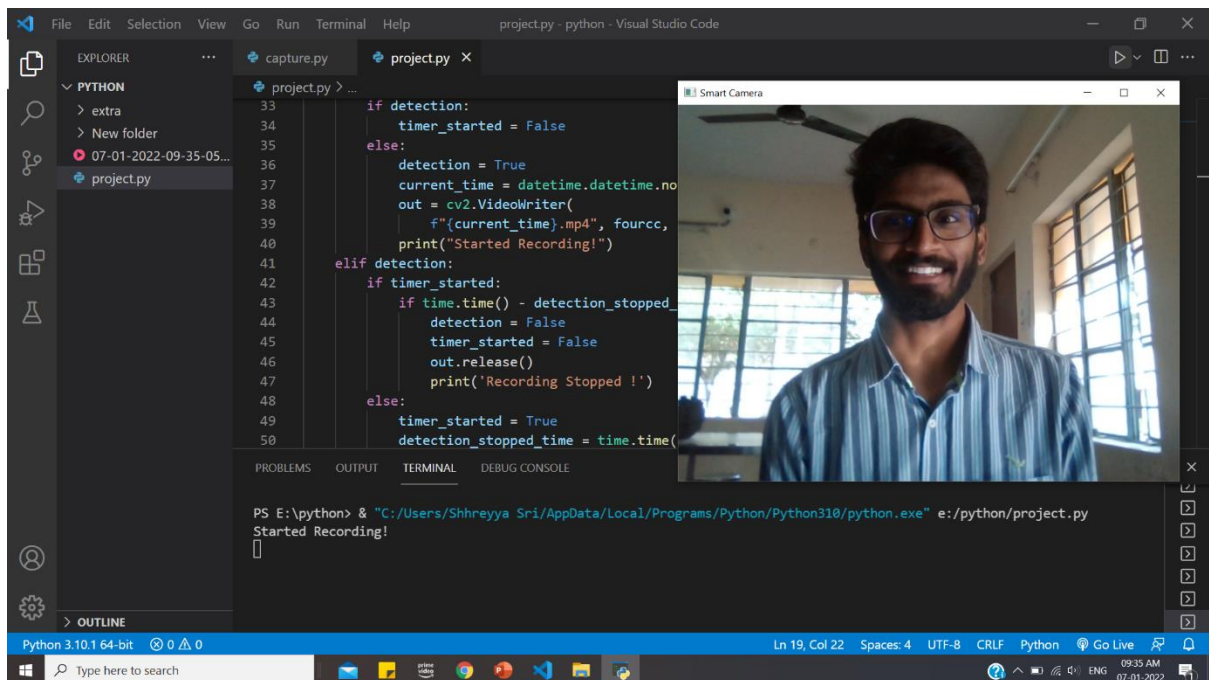

```
if detection:
    out.write(frame)
cv2.imshow("Smart Camera", frame)
```

In this part of code frames are added to recording when the detection is true. `cv2.imshow()` method is used to display an image in a window and window is named as smart camera.

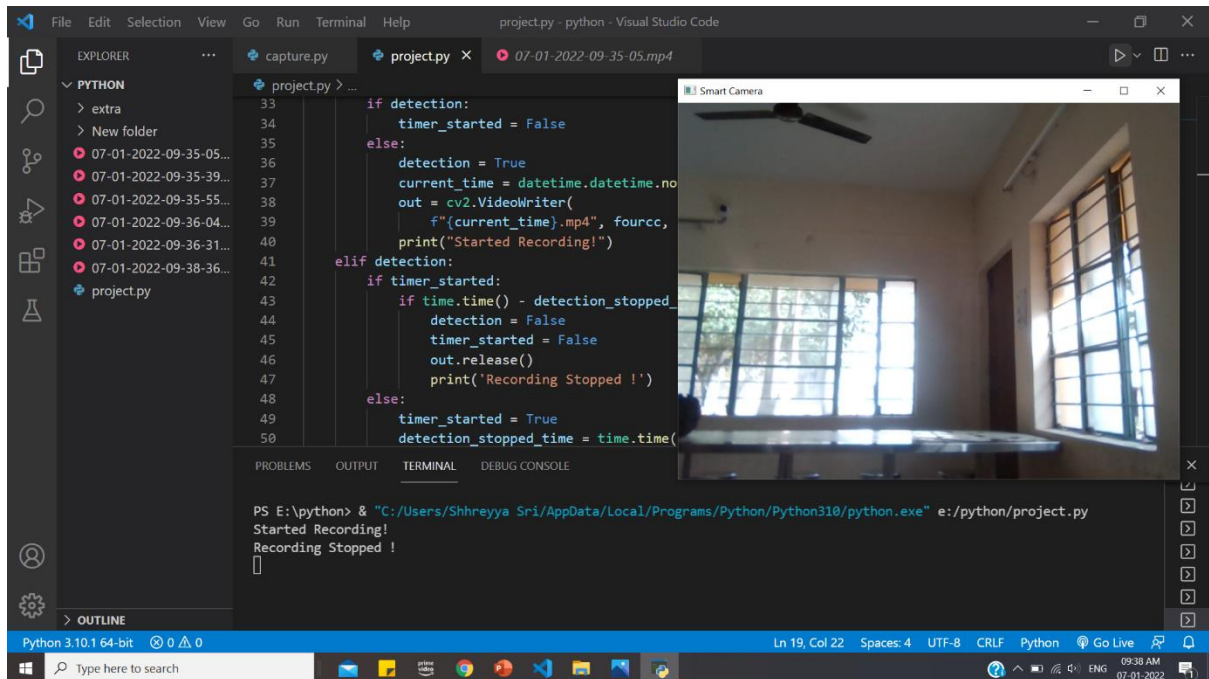
```
if cv2.waitKey(1) == ord('q'):
    break
out.release()
cap.release()
cv2.destroyAllWindows()
```

This if statement checks the key entered by the user. If the user enters character 'q', the execution ends as there is a break statement and comes out of while condition. We use `waitkey()` function to pause frame in the video. This piece of code will stop the program when you press q on the keyboard. We just need to turn off the camera and close the window when we stop the program. So we use `release()` function to turn off the camera and `destroyAllWindows()` to close the window.

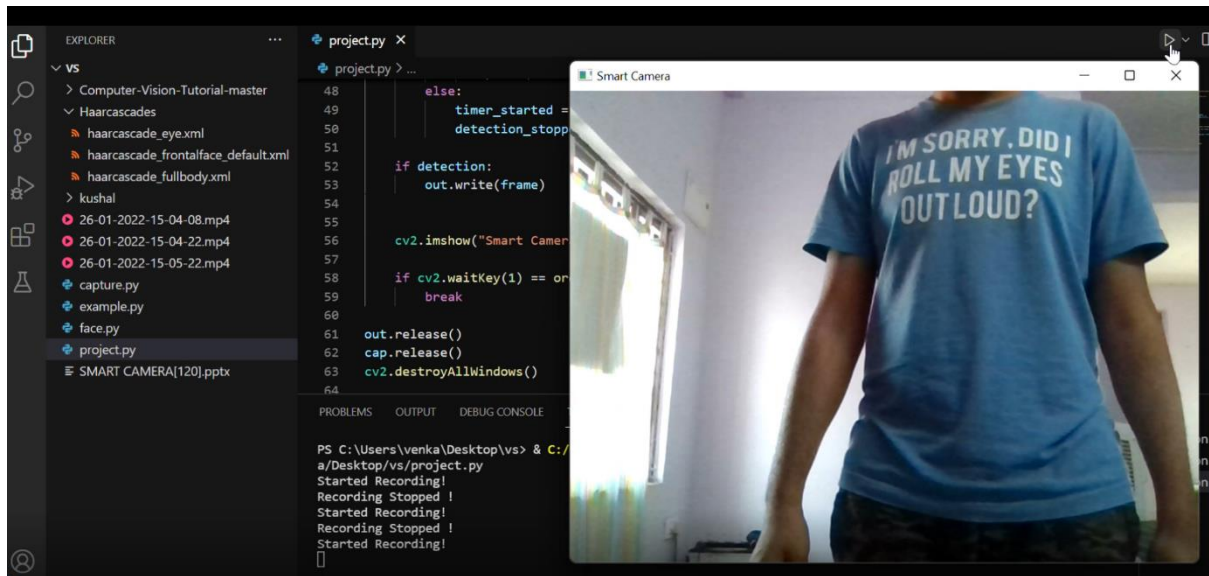
TEST CASES / RESULTS



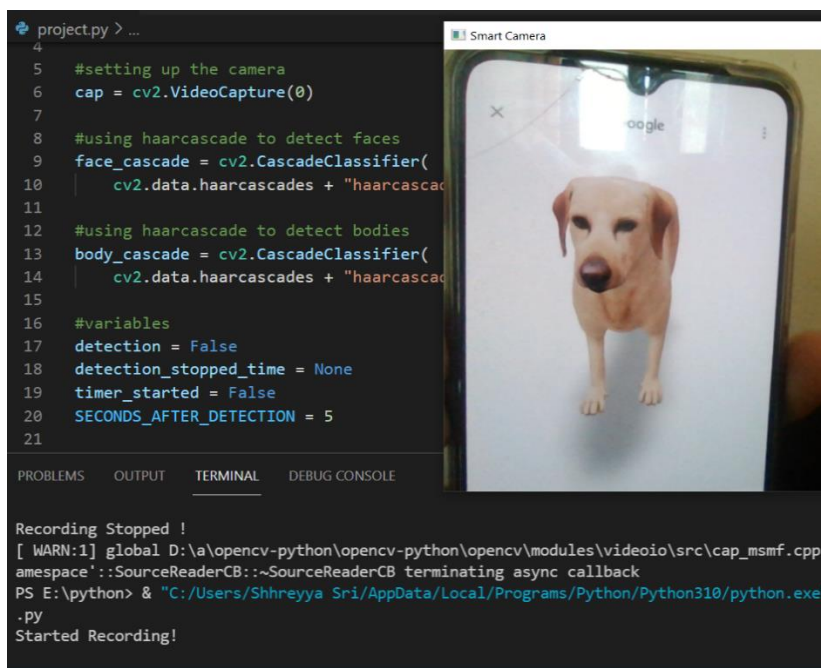
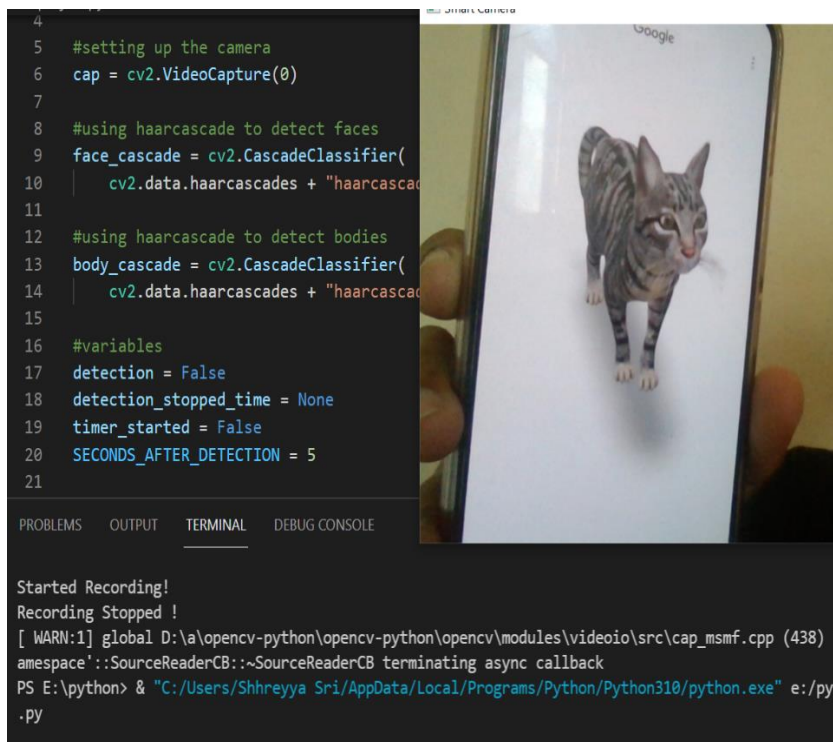
Person is detected when he is Infront of smart camera. You can see the recording has been stored with date and time. In the terminal we got a notification that recording as been started.



When there is no person the recording has been stopped and the video is saved in the local folder. We can see a notification in terminal saying the recording has been stopped.





As u can see the person face is not there in the camera vision, yet the recording has been started, as we used full body detection the body is getting detected.





In case of animals, sometimes it is being detected and mostly its not detected. Sometimes it is detected due to eyes.

- Also identifies a person with mask.
- If we show a picture of a person also it identifies and starts recording.
- Body detection is not so accurate.
- Mostly cant detect animals , but in few cases due to eyes it is detected as a human.

STORAGE

 WIN_20220119_1...		19-01-2022 06:23 PM	MP4 File	43,767 KB	00:00:46
---	---	---------------------	----------	-----------	----------

The above is the space consumed by recordings of normal camera, i.e.
43,767 KB

 19-01-2022-18-24...	19-01-2022 06:24 PM	MP4 File	4,334 KB	00:00:13
 19-01-2022-18-25...	19-01-2022 06:25 PM	MP4 File	5,017 KB	00:00:14

The above is the space consumed by recordings of smart camera,
i.e. $4,333 + 5,017 = 9,350$ KB
Space saved = 34,417 KB.

FUTURE SCOPE

The next advancement of our project would be a motion detection. The recording would be done when there is a motion in the environment. It could be implemented as, whenever there is a difference in frame (motion takes in environment) the video starts recording and records until the present frame matches with the first frame.

This advancement would overcome the drawback of our project as it could not detect animal or any other object.

CONCLUSION

With this project, we can increase the surveillance system at a very low cost and easy implementation. It also reduces the storage by capturing only the necessary things. Smart camera can be used almost everywhere.