PROJECT REPORT

Date	20 MAY 2023
Team ID	NM2023TMID01052
Project Name	Project- AI Enabled Car Parking using Open CV
Team Members	L.SHANGEETHA R.DEVIKA S.DHARSHINI V.DHARSHINI

TABLE OF CONTENTS 1. INTRODUCTION

Car parking is a common problem faced by drivers in busy urban areas. For example, imagine you are driving to a shopping mall during peak hours. As you approach the mall, you notice that the parking lot is full, and several other cars are circling around looking for available spots. You join the queue of cars, hoping to find an available spot soon. However, as time passes, you realize that the parking lot is overcrowded, and it's becoming increasingly difficult to find a spot. You start to feel frustrated and anxious, knowing that you might be late for your appointment or miss out on a great shopping opportunity.

1.2 PURPOSE

AI-enabled car parking using OpenCV is a computer vision-based project that aims to automate the parking process. The project involves developing an intelligent system that can identify empty parking spaces and it gives the count of available parking spots. The system uses a camera and OpenCV (Open Source Computer Vision) library to capture live video footage of the parking

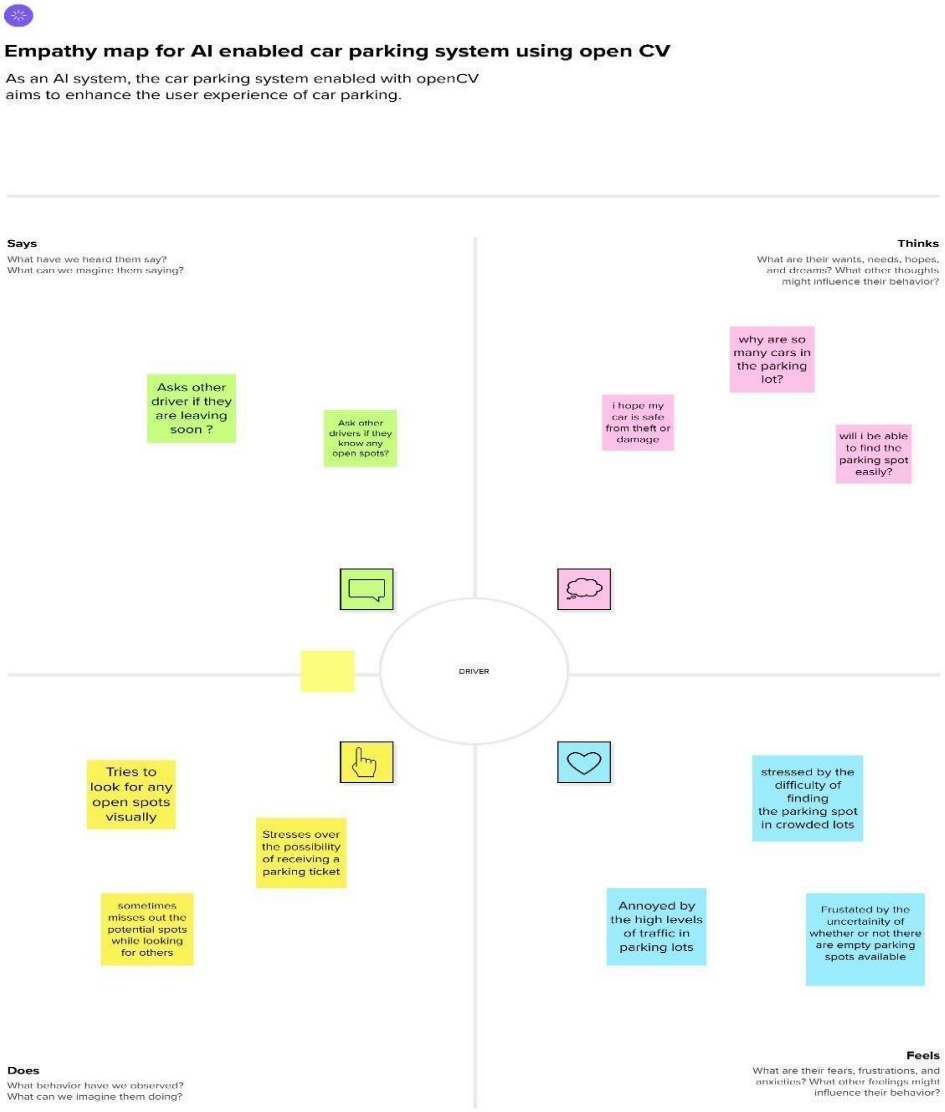
2.IDEATION AND PROPOSED SOLUTION

2.1 PROBLEM STATEMENT DEFINITION

Develop an AI-enabled car parking system using OpenCV that can accurately detect and track vehicles entering and exiting a parking lot, and provide real-time information about available parking spaces.

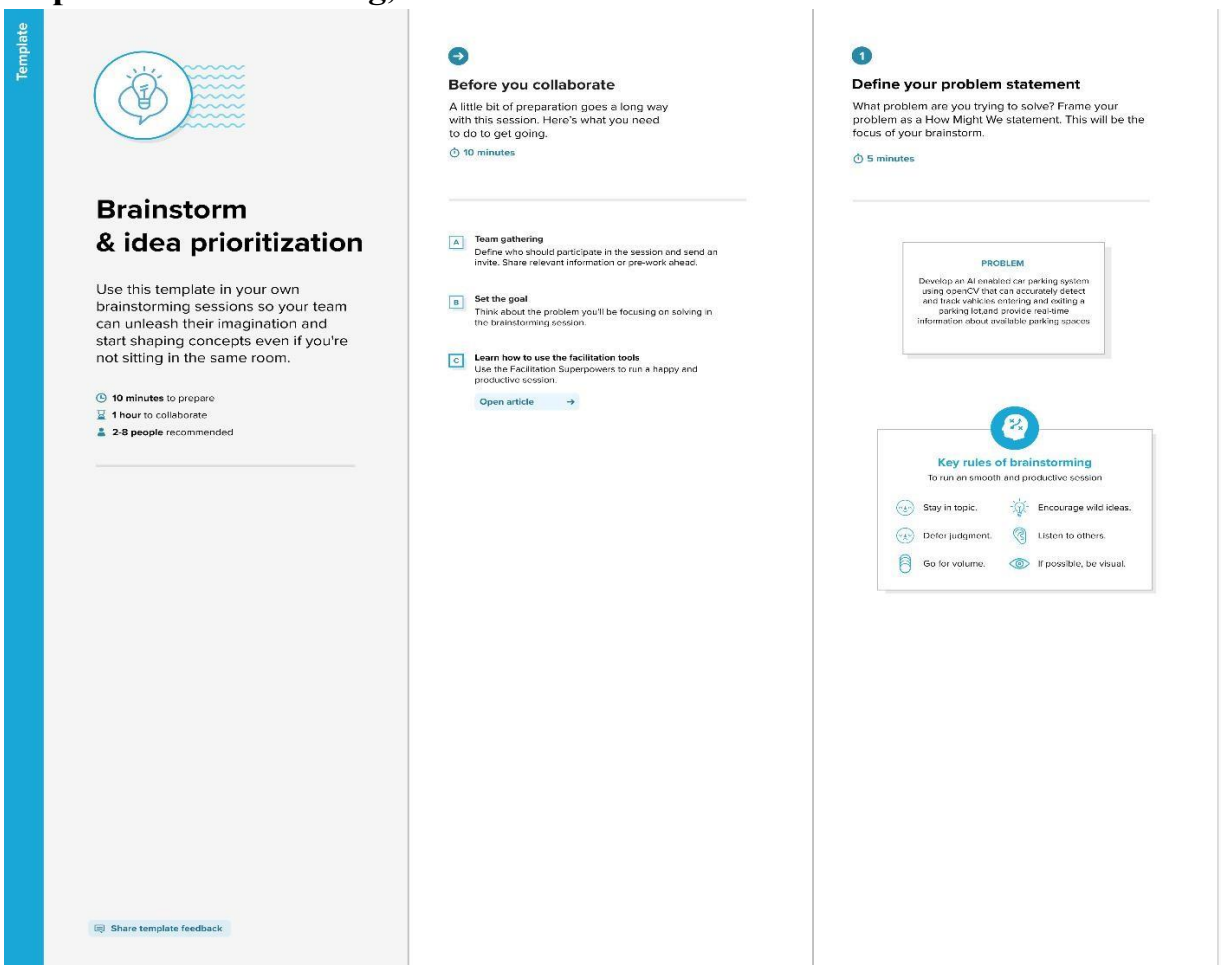
AI-enabled car parking using OpenCV is a computer vision-based project that aims to automate the parking process. The project involves developing an intelligent system that can identify empty parking spaces and it gives the count of available parking spots. The system uses a camera and OpenCV (Open Source Computer Vision) library to capture live video footage of the parking lot.

2.2 EMPATHY MAP CANVAS

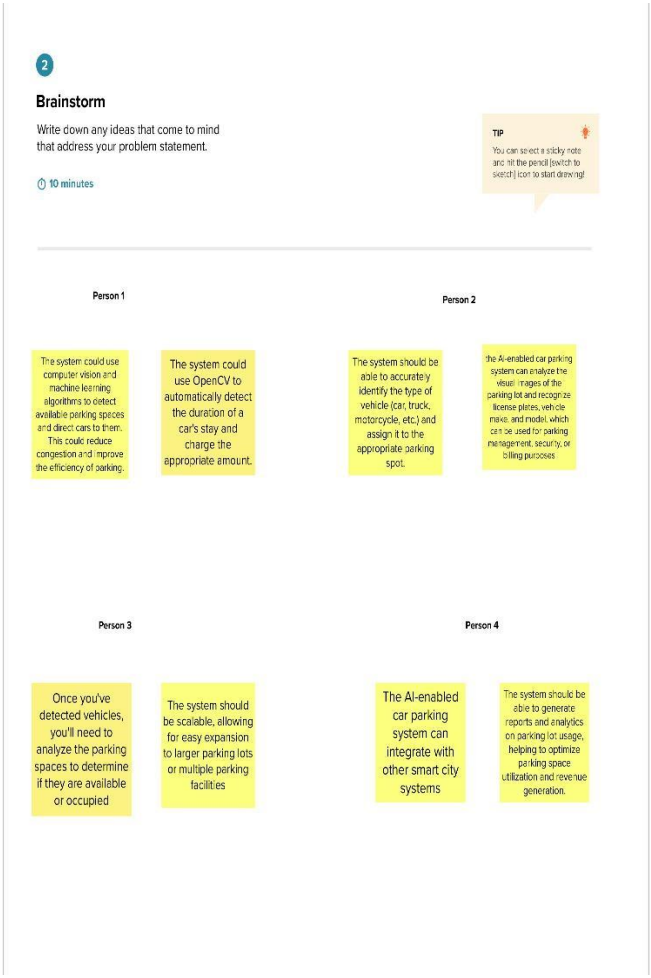


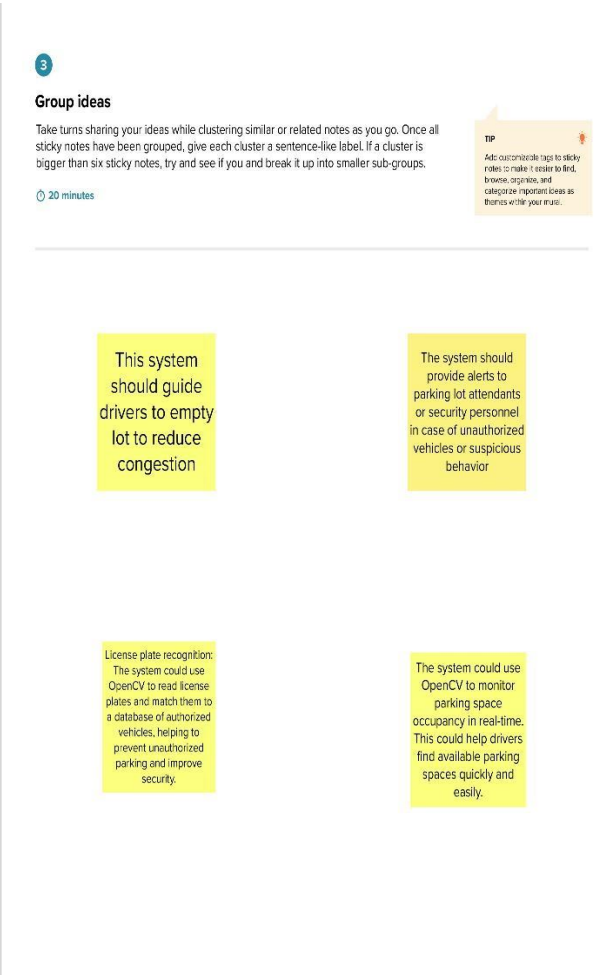
2.3 IDEATION AND BRAINSTORMING

Step-1: Team Gathering, Collaboration and Select the Problem Statement

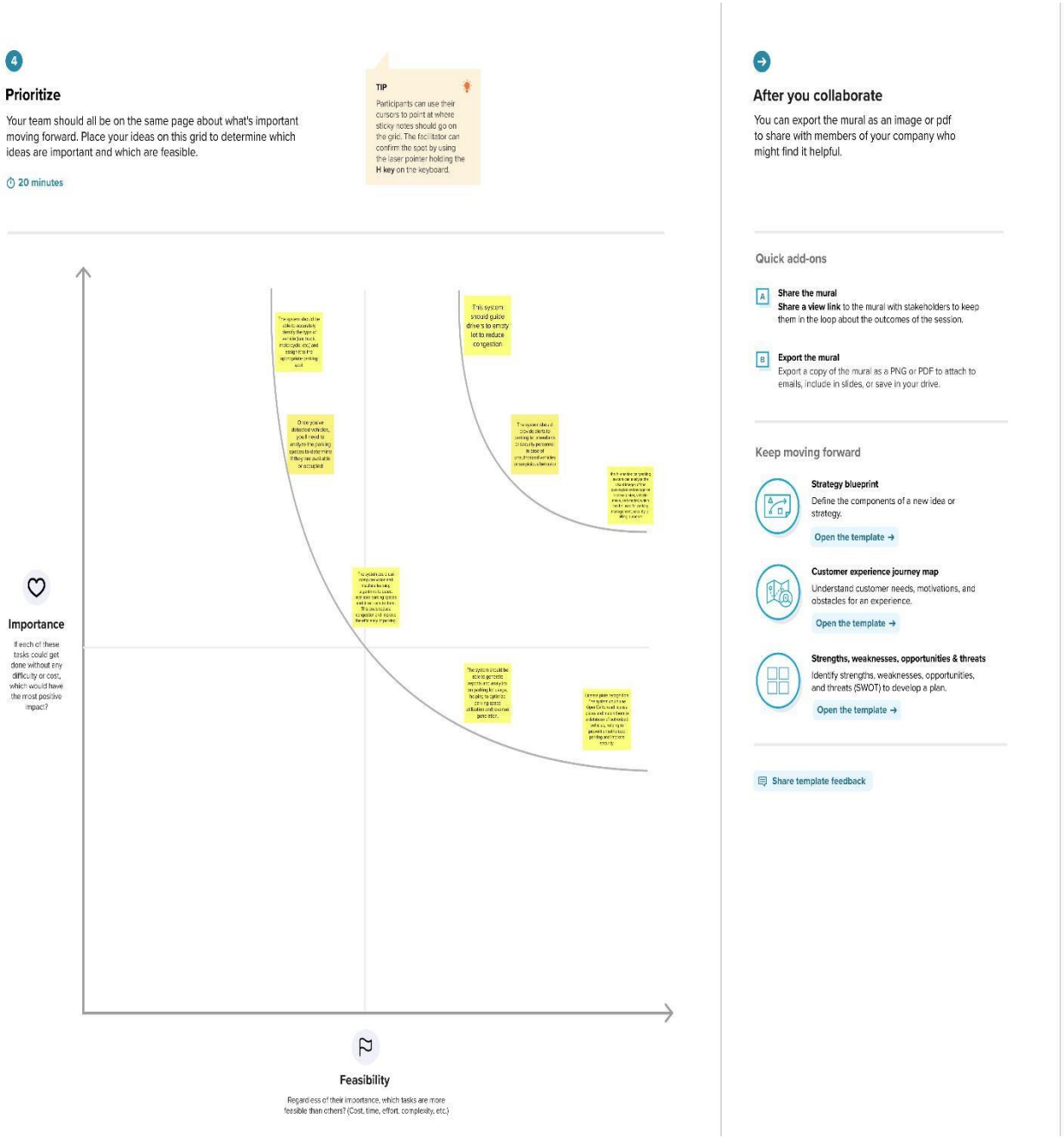


Step-2: Brainstorm, Idea Listing and Grouping





Step-3: Idea Prioritization



2.4 PROPOSED SOLUTION

S.NO	Parameter	Description
1.	Problem Statement	Develop an AI-enabled car parking system using OpenCV that can accurately detect and track vehicles entering and exiting a parking lot, and provide real-time information aboutavailable parking spaces.

2.	Idea/Solution description	AI-based smart parking is an innovative parking solution that leverages data from different devices like sensors and cameras to form an AIdriven parking management system to detect the availability of parking spots.
3.	Noveity/Uniqueness	Inherent safety and security Compared to conventional parking garages, Automated Parking Systems are inherently much safer and more secure because they remove driving and pedestrians from the parking area. No driving means no car damage or possibility of stolen cars.
4.	Social Impact/Customer Satisfaction	Reduce search traffic for parking Smart parking helps combat this problem by reducing the number of vehicles driving slowly around the city looking for parking spaces. This ensures proper traffic flow, reducing congestion in cities with limited parking spaces.
5.	Business Model(revenue model)	AI-enabled car parking system using OpenCV can provide an efficient solution for car parking management, helping to optimize the usage of parking spaces, improve customer experience, and increase revenue for parking lot owners.
6.	Scalability of the solution	The AI-enabled car parking system using OpenCV is highly scalable and can be customized to accommodate parking lots of various sizes and types

3.REQUIREMENT ANALYSIS

.

3.1 Functional Requirements:

Following are the functional requirements of our proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	**load the video**	load the appropriate parking lot video.
FR-2	**Analyse of video**	Preparation of raw data and make it suitable for building of machine learning model.

FR-3	**Building Artificial Intelligence model**	✓ Load the video in the model ✓ Determine the machine learning techniques that will be used to train the AI model. ✓ Process the video and get output ✓ Deploy the model
FR-4	**Train the data**	Train the model using training video.
FR-5	**Test the data**	At last, test the model for evaluation of final model.

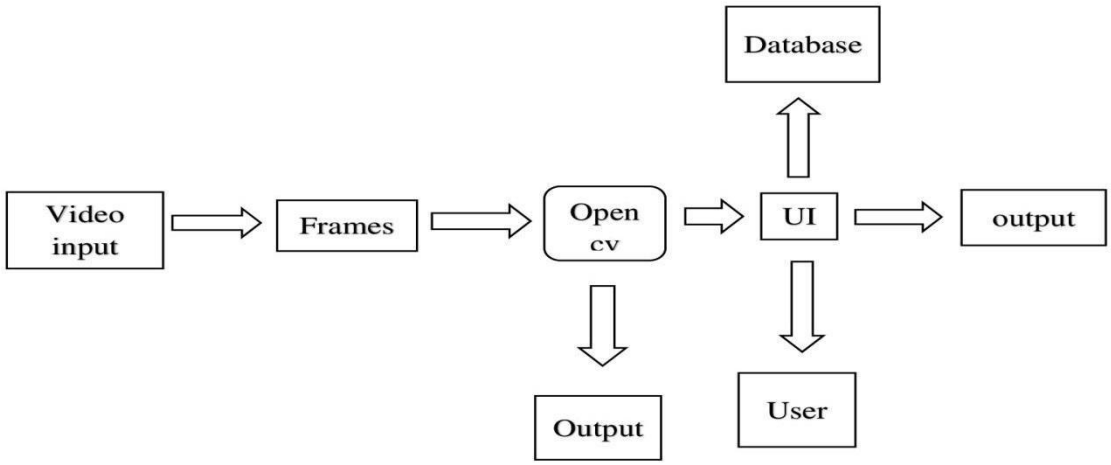
3.2 Non-functional Requirements:

Following are the non-functional requirements of our proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	It can be use by all the drivers in the parking lot to know the empty
NFR-2	**Security**	Providing secure system to all the users who are all using the parki
NFR-3	**Reliability**	System will operate without failure for a specific period of time.
NFR-4	**Performance**	Our model predictions are same as the true values. So, the performa
NFR-5	**Availability**	Available to different group of companies which has largest parkin
NFR-6	**Scalability**	In our model, Prediction of parking lot will be faultless.

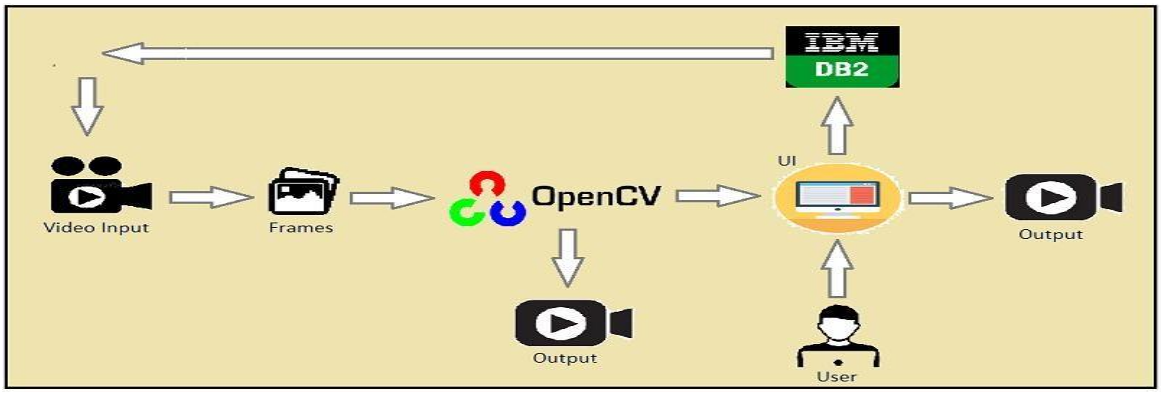
4.PROJECT DESIGN

4.1 DATA FLOW DIAGRAM



4.2 SOLUTION AND TECHNICAL ARCHITECTURE

Technical Architecture



4.3 USER STORIES

User type	User story number	User story/task	Team member
DRIVER	USN-1	As a driver, I am able to find a parking spot quickly and easily using an AIenabled car parking system so that I can	**Shangeetha.L**

		save time and reduce frustration	
USER	USN-2	As a car park user, AI-enabled car parking system is able to provide me with real-time information on available parking spots and their locations using OpenCV	**Devika.R**
OWNER	USN-3	As a car park owner, I am able to identify parking violations automatically using OpenCV, such as parking in a disabled spot, so that I can enforce parking regulations and maintain safety	**Dharshini.S**
MANAGER	USN-4	As a car park manager, I am able to detect the number of available parking spots in real-time using OpenCV so that I can manage the parking lot more efficiently	**Dharshini.v**

5.CODING AND SOLUTIONING

5.1 FEATURE 1

TRAFFIC FLOW OPTIMIZATION

By monitoring the parking area, the system can analyze the flow of vehicles and identify congested areas or potential bottlenecks. It provides valuable insights to parking attendants or traffic management personnel, enabling them to take proactive measures to alleviate congestion and improve traffic flow.

PARKING SPACE MONITORING

The system analyzes the video streams to determine the occupancy status of each parking space. It can differentiate between vacant and occupied spots, providing real-time updates on parking availability to drivers

6.RESULTS

6.1 PERFORMANCE METRICES

S.No	Parameter	Values
1.	Model Summary	In this model, it has **Data Acquisition**, **Object Detection, Vehicle Tracking**, **Parking Space Detection** and **Occupancy detection** through sensors.
2.	Accuracy	Training Accuracy – **92%** Validation Accuracy **-94%**
3.	Confidence Score (Only Yolo Projects)	Class Detected Confidence Score – **"if count < 900"**

8.CONCLUSION

This study's main beneficence is to perfect the unearthing of open parking spaces in an expenditure to ease parking arena slowdown. The development of machine learnedness and vision- grounded technology has made it possible for motorcars to find open spaces at parking lots using affordable automatic parking systems. unborn studies can concentrate on assigning specific emplacements to customers who have afore registered with an online parking management system. The precision about the proposal algorithm is inaugurated to be 92.The outcomes demonstrates that, when the captured photos of the parking lot aren`t clear due to low lighting or overlaps, the productivity drops and the exactitude for spotting decreases. It's noticed that the average performance is 99.5 and is remarkably high as contrasted with other parking lot finding out procedures. The effectiveness of the proposed method in some cases drops down due to the strong darkness. The ultra precision of Get image frames RGB to Gray image Do Calibration Get equals of parking spot Get fellows of car Parking spot divided into Blocks Convert Block to inverse binary Get value of connected locality to determine autos number of free and Reserved Blocks Input Live stream recording 1313 the proposed task additionally relies on the kind of camera utilized for covering the parking lot.

9.FUTURE SCOPE

- Hook up a webcam to a snort Pi and have live parking monitoring at home
- It's effective at resolving parking issues. In addition, it provides automatic billing, as well as eliminating traffic congestion. Utilizing a multilevel parking technique, this work can be further developed into a fully automated system.
- The system presents the details of vacant parking areas nearby, and reduces the market problems related to illegal parking in the area. It was intended to meet the requirements of controlled parking that offers downhill parking techniques to the authorities.

10. APPENDIX

10.1 SOURCE CODE

```
#main.py import cv2
    import pickle
    import cvzone
    import numpy as np

    # Video feed
    cap = cv2.VideoCapture('carPark.mp4')

    with open('CarParkPos', 'rb') as f:
       posList = pickle.load(f)
    width, height = 107, 48
    def checkParkingSpace(imgPro):
       spaceCounter = 0
        for pos in posList:
          x, y = pos

          imgCrop = imgPro[y:y + height, x:x + width]
          # cv2.imshow(str(x * y), imgCrop) count =
          cv2.countNonZero(imgCrop)



          if count < 900: color
          = (0, 255, 0)
          thickness = 5
          spaceCounter += 1
          else:
             color = (0, 0, 255)
             thickness = 2
```

```
            cv2.rectangle(img, pos, (pos[0] + width, pos[1] + height), color, thickness)
            cvzone.putTextRect(img, str(count), (x, y + height - 3), scale=1,
            thickness=2, offset=0, colorR=color)

        cvzone.putTextRect(img, f'Free: {spaceCounter}/{len(posList)}', (100, 50), scale=3,
                           thickness=5, offset=20, colorR=(0,200,0))
    while True:
        if cap.get(cv2.CAP_PROP_POS_FRAMES) ==
    cap.get(cv2.CAP_PROP_FRAME_COUNT):
            cap.set(cv2.CAP_PROP_POS_FRAMES, 0)
        success, img = cap.read()
        imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        imgBlur = cv2.GaussianBlur(imgGray, (3, 3), 1)
        imgThreshold=cv2.adaptiveThreshold(imgBlur, 255,
    cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                           cv2.THRESH_BINARY_INV, 25, 16)
        imgMedian = cv2.medianBlur(imgThreshold, 5)
        kernel = np.ones((3, 3), np.uint8) imgDilate =
        cv2.dilate(imgMedian, kernel, iterations=1)
        checkParkingSpace(imgDilate)
cv2.imshow("Image", img) #
cv2.imshow("ImageBlur", imgBlur) #
cv2.imshow("ImageThres", imgMedian)
cv2.waitKey(10) #main_trackers.py

import cv2 import
pickle import
cvzone import
numpy as np

cap = cv2.VideoCapture('carPark.mp4')
width, height = 103, 43 with
open('polygons', 'rb') as f:
    posList = pickle.load(f)


def empty(a):
    pass
```

```
cv2.namedWindow("Vals") cv2.resizeWindow("Vals",
640, 240) cv2.createTrackbar("Val1", "Vals", 25, 50,
empty) cv2.createTrackbar("Val2", "Vals", 16, 50,
empty) cv2.createTrackbar("Val3", "Vals", 5, 50,
empty)


def checkSpaces():
    spaces = 0 for pos in
    posList: x, y = pos w,
    h = width, height

        imgCrop = imgThres[y:y + h, x:x + w]

        count = cv2.countNonZero(imgCrop)

        if count < 900:

            color = (0, 200,
            0) thic = 5 spaces
            += 1


        else:
            color = (0, 0, 200) thic = 2 cv2.rectangle(img, (x,

        y), (x + w, y + h), color, thic)

        cv2.putText(img, str(cv2.countNonZero(imgCrop)), (x, y + h - 6),
cv2.FONT_HERSHEY_PLAIN, 1,
                    color, 2)

    cvzone.putTextRect(img, f'Free: {spaces}/{len(posList)}', (50, 60), thickness=3, offset=20,
                       colorR=(0, 200, 0))


while True:

    # Get image frame
    success, img = cap.read()
```

```
    if     cap.get(cv2.CAP_PROP_POS_FRAMES)                                    ==
cap.get(cv2.CAP_PROP_FRAME_COUNT):
        cap.set(cv2.CAP_PROP_POS_FRAMES, 0)
    # img = cv2.imread('img.png')
    imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    imgBlur = cv2.GaussianBlur(imgGray, (3, 3), 1)
    # ret, imgThres = cv2.threshold(imgBlur, 150, 255, cv2.THRESH_BINARY)

    val1 = cv2.getTrackbarPos("Val1", "Vals")
    val2 = cv2.getTrackbarPos("Val2", "Vals")
    val3 = cv2.getTrackbarPos("Val3", "Vals")
    if val1 % 2 == 0: val1 += 1
    if val3 % 2 == 0: val3 += 1

    imgThres                =                cv2.adaptiveThreshold(imgBlur,                255,
cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                            cv2.THRESH_BINARY_INV, val1, val2)
    imgThres = cv2.medianBlur(imgThres, val3)
    kernel = np.ones((3, 3), np.uint8) imgThres =
    cv2.dilate(imgThres, kernel, iterations=1)
    checkSpaces() #
    Display Output

    cv2.imshow("Image", img) #
cv2.imshow("ImageGray", imgThres) #
cv2.imshow("ImageBlur", imgBlur) key =
cv2.waitKey(1) if key == ord('r'): Pass
```

#parkingspacepicker.py

```
import cv2 import
pickle width, height =
107, 48

try:
    with open('CarParkPos', 'rb') as f:
        posList = pickle.load(f)
except:
    posList = []
```

```
def mouseClick(events, x, y, flags, params): if
   events == cv2.EVENT_LBUTTONDOWN:
   posList.append((x, y)) if events ==
   cv2.EVENT_RBUTTONDOWN: for i, pos in
   enumerate(posList):
         x1, y1 = pos if x1 < x < x1 + width and y1 < y
         < y1 + height: posList.pop(i)

   with open('CarParkPos', 'wb') as f:
      pickle.dump(posList, f)
while True:
   img = cv2.imread('carParkImg.png') for pos in posList: cv2.rectangle(img,
   pos, (pos[0] + width, pos[1] + height), (255, 0, 255), 2)

   cv2.imshow("Image", img)
   cv2.setMouseCallback("Image", mouseClick)
   cv2.waitKey(1)
```

10.2 Github and project video demo link Github link:

https://github.com/naanmudhalvan-SI/PBL-NT-GP--4892-1680773584.git

Video demo link:

https://drive.google.com/file/d/1vAy7LQLByJl3jpOIUP2w3czctHiSE2C/view?usp=drivesdk