ABSTRACT

The problem of finding an appropriate parking space is a challenging one, particularly in large cities. With the increase in car ownership, parking spaces have become scarce. The growing demand for these spots coupled with limited availability has led to imbalances between supply and demand. A lack of adequate parking management systems has resulted in many streets being littered with illegally parked cars. A scalable, reliable, and efficient parking management system is needed to combat this problem. Deep learning-based computer vision techniques have emerged as promising solutions for such problems. These technologies have had a huge impact on the field of image recognition and processing. They also present great potential for further applications in the area of vehicle tracking. Hence, they can be used to detect parking spots.

A densely packed city center can be an unbearable place to park your car. Finding parking spaces can prove frustrating if you're not careful. Automatic smart parking systems promise to ease the burden of finding a spot in busy areas. To help drivers find a parking spot, we have developed a vision-based smart-parking framework. First, we divided the parking lot into blocks and categorized each block to determine whether it was occupied or empty. Then we sent information about the availability of free or reserved parking to motorists on their smartphones. Our system demonstrates superior performance compared to commercially available solutions because it offers higher accuracy.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

Abbreviation	Definition
Open CV	Open-Source Computer Vision Library
CCTV	Closed-Circuit Television
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
MySQL	My Structured Query Language

I.INTRODUCTION

Most parking lots today are still managed by hand. There is no automated monitoring system in place to keep track of how much capacity each parking place contains. In order to find an empty spot, drivers often have to make a circuitous trip through the parking lot. Where there are more people than parking spots, such problems are especially common near hospitals, malls, schools, and other large gathering places.

The process of finding a free parking space can take a lot of time and involve driving around in circles. These days, parking spots are often occupied so badly that they're almost unusable. Poorly managed parking areas lead to inefficient utilization of the parking spaces. This causes a lot of traffic jams near the parking areas.

We propose a new method to improve the efficiency of parking lots by counting how much space is left in each parking zone and displaying that information to drivers via a smartphone app. We employ a camera to photograph the parking lot and use image processing approaches to determine if any vehicles are parked in each section. Whenever a vehicle moves into or out of a particular parking zone, the status of the whole lot changes.

II. LITERATURE REVIEW

In this paper , for an autonomous vehicle parking system , we have created and implemented a framework. The experimental results demonstrated the suggested system's ability to provide accurate data. The issue of parking in crowded regions has been addressed using a variety of strategies and forms. A approach for counting the cars at the checkpoint from which the variety of open parking spots can be counted was once proposed via Ming-Yee Chiu et al. Induction loop sensors are positioned below the road surface to elevate out the counting. Although the usage of sensors was once much less highly-priced, they aren't fluently told by environmental factors, and they reliably detect, their installation was challenging and resulted in road damage. In the event of a problem, maintenance was extremely grueling .Grounded on vision-based techniques, various categories of discovery styles are described. The entire parking lot that's open for parking can be analyzed by the camera using vision-grounded ways; the facts is also analyzed, and the output will specify the unique quantity and regions of the open parking spaces. According to Zhang Bin et al., vision- grounded parking spot detection techniques are relatively simple to set up, inexpensive, and the detector can be quickly modified to meet needs. also, the information gleaned from photographs is quite rich. The precision of the vision approach is severely reliant on the camera's position.

III.METHODOLOGY

Videos were recorded using a camera that was ten feet above the parking lot. In order to ameliorate the system's ability to recognize objects, video footage was collected under various environmental and temporal situations. Frames are used to segment video. also, to reduce computational complexity, a key frame is uprooted from each segment and subjected to additional processing. Key frame subtraction is used to estimate the motion of the toy auto when it enters or exits the parking lot from the parking arena.

At first, there were no parking lanes in the parking lot. The user must manually enter the location of the intended parking spot and the car. The system automatically creates virtual parking spaces while taking the size of the vehicle into consideration. In our training model, the number of parking spaces is limited to fourteen. Each parking lot has a different numeric label.

Our system will check to see if there are any cars in each block after the parking area has been partitioned into virtual blocks.

Inverse binary is used to take out the car as the area of past time ROI after applying a binary filter to the image. Calculating the connected region's value in ROI and designating a parking space as reserved when the threshold value exceeds eighty. The count of unreserved sections will be displayed to drivers in green, while the number of reserved sections will be displayed in red.

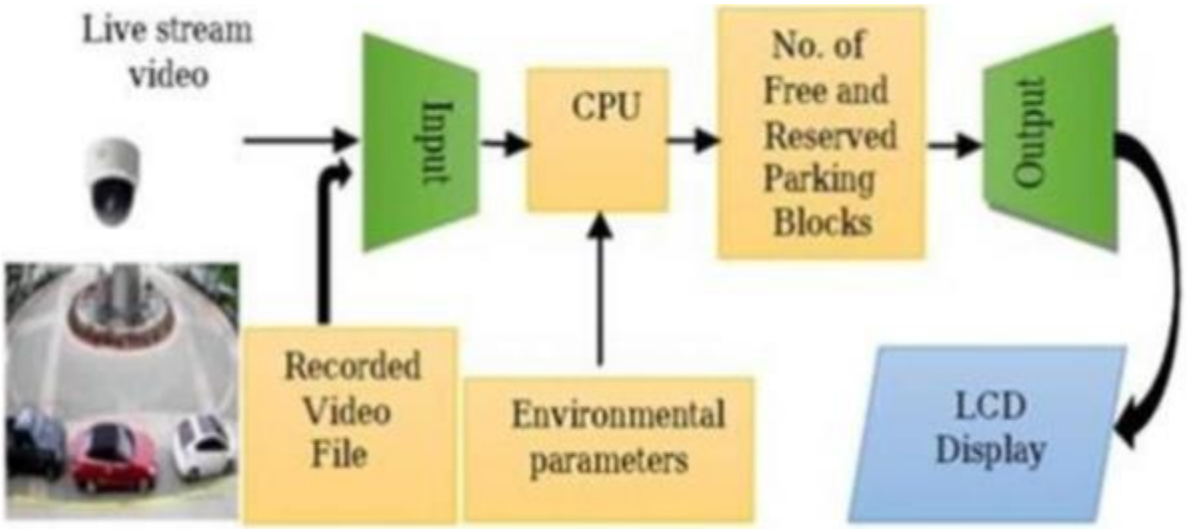


Figure 1. Block diagram of proposed system

Labeling of a detection position is initial step. Labeled scenes include:

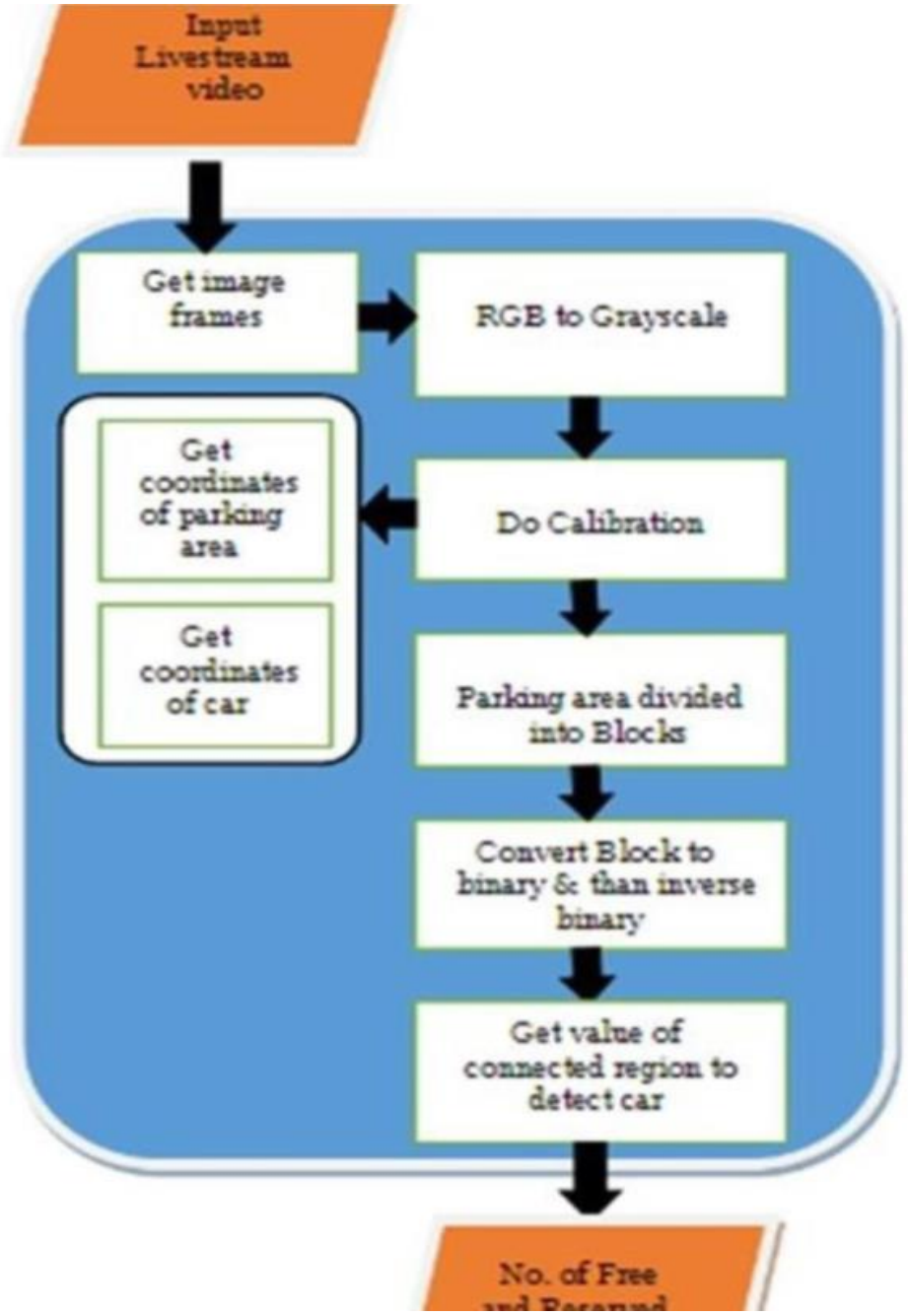
Figure 2.Parking slot

Annotated pictures:

Figure 3.Parking place detection

This will identify each parking place for detection.

VI.DESIGN AND IMPLEMENTATION



The primary route-way of the proposed algorithm for parking space discovery are:

1. The parking lot will be live- streamed by the camera to the system.
2. When a horseless carriage pulls into or out of the parking space, filmmaking are taken
3. Gray scale images are created by converting RGB images
4. Make adjustments

   Choosing the parking lot's equals first is a good idea. This will remove any unnecessary white space from the image other than the parking lot.

   • Next, decide where the single parking space's parallels are. As a result, the parking lot will be divided into spaces of cognate size.
5. In order to turn the parking lot into black and the auto into white, each block is first converted from gray scale to double and then to inverse binary
6. To determine if a block contains a car or not, a threshold value is computed for each block. Blocks are free and available for parking if their value is less than a threshold value, and they're occupied if their worth exceeds the threshold.

V.PLATFORMS USED

1. Open CV- python Open CV, a sizable open- source library for computer vision, machine erudition, and image processing, currently plays a significant part in real- time operation, which is crucial in modern systems. It can be used to process pictures and pictures so that people can fete goods, people's faces, and even human handwriting.
2. The Open CV array structure can be reclaimed by Python for figure out when it's coupled with a variety of libraries, cognate as Num Py. We use vector space and implement fine activities to the aspects of a visual pattern in order to identify it.
3. Spot Pickle is mostly used in Python to serialize and non sequential Python object formats. To place it different way, it’s the procedure about converting a Python object into a byte run so that it can be saved in a column or memory, have its state preserved across sessions, or be used to transfer data over a network. By using the pickled byte stream and besides un pickling it, the original object hierarchy can be recreated. Object serialization in Javaor.Net is correspondent to this entire process.
4. CV zone : This computer vision package facilitates the implementation of AI and image processing operations. It primarily makes use of the OpenCV and Media pipe libraries.
5. NumPy : For the Python programming language, NumPy is a library that adds support for largish , multi-layered arrays and matrices. It additionally affords a big count of high-stage mathematical features to work with these arrays.
6. Open CV has a function called cv2 that can read tape recording(). Pass 0 in the function parameter allows us to penetrate our webcam. The RTSP create a satisfactory terrain for productive Python, web, and statistics knowledge development url can be dispatched as a Pycharm or other IDE

7. PyCharm is a specialized Python Integrated Development Environment(IDE) that gives a vast range of crucial equipment for Python developers. These equipments are tightly built in to cv2 library.

IV.CODES AND OUTPUTS

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)
```

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)
```

Index.html

```
<!DOCTYPE html>
<!-- Coding By CodingNepal - codingnepalweb.com -->
<html lang="en">
```

```
<head>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Website with Login & Registration Form</title>
  <link rel="stylesheet" href="Styles/style.css" />
  <!-- Unicons -->
  <link rel="stylesheet" href="https://unicons.iconscout.com/release/v4.0.0/css/line.css" />
</head>
<body>
  <!-- Header -->
  <header class="header">
    <nav class="nav">
      <a href="#" class="nav_logo">AI Enabled Car Parking </a>

      <ul class="nav_items">
        <li class="nav_item">
          <a href="#" class="nav_link">Home</a>
          <a href="#" class="nav_link">Model</a>
          <a href="#" class="nav_link">Contact</a>
          <a href="#" class="nav_link">About</a>
        </li>
      </ul>

      <button class="button" id="form-open">Login</button>
    </nav>
  </header>

  <!-- Home -->
  <section class="home">
    <div class="form_container">
      <i class="uil uil-times form_close"></i>
      <!-- Login From -->
      <div class="form login_form">
        <form action="#">
          <h2>Login</h2>
```

```
    <div class="input_box">
      <input type="email" placeholder="Enter your email" required />
      <i class="uil uil-envelope-alt email"></i>
    </div>
    <div class="input_box">
      <input type="password" placeholder="Enter your password" required />
      <i class="uil uil-lock password"></i>
      <i class="uil uil-eye-slash pw_hide"></i>
    </div>

    <div class="option_field">
      <span class="checkbox">
        <input type="checkbox" id="check" />
        <label for="check">Remember me</label>
      </span>
      <a href="#" class="forgot_pw">Forgot password?</a>
    </div>

    <button class="button">Login Now</button>

    <div class="login_signup">Don't have an account? <a href="#" id="signup">Signup</a></div>
  </form>
</div>

<!-- Signup From -->
<div class="form signup_form">
  <form action="#">
    <h2>Signup</h2>

    <div class="input_box">
      <input type="email" placeholder="Enter your email" required />
      <i class="uil uil-envelope-alt email"></i>
    </div>
    <div class="input_box">
      <input type="password" placeholder="Create password" required />
```

```
            <i class="uil uil-lock password"></i>
            <i class="uil uil-eye-slash pw_hide"></i>
          </div>
          <div class="input_box">
            <input type="password" placeholder="Confirm password" required />
            <i class="uil uil-lock password"></i>
            <i class="uil uil-eye-slash pw_hide"></i>
          </div>

          <button class="button">Signup Now</button>

          <div class="login_signup">Already have an account? <a href="#" id="login">Login</a></div>
        </form>
      </div>
    </div>
  </section>

  <script src="Styles/script.js"></script>
 </body>
</html>
```

Model.html

```
<!DOCTYPE html>
<!-- Coding By CodingNepal - codingnepalweb.com -->
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta http-equiv="X-UA-Compatible" content="ie=edge">
 <title> Access The parking lot </title>
 <!---Custom Css File!--->
 <link rel="stylesheet" href="Styles/style1.css">
</head>
<body>
```

```
  <section class="about-us">
    <div class="about">
      <!--<img src="girl.png" class="pic">-->
      <div class="text">
        <h2>Parking Lot</h2>
        <h5>Access the parking lot</h5>
          <p>Parking lots provide a convenient and safe place for people to park their vihicles while they engage in various avtivities.You touch the <b>Click To View button</b> to view the parking spaces</p>
        <div class="data">
        <a href="/Mini Project/main.py" class="hire">Click to View</a>
        </div>
      </div>
    </div>
  </section>
</body>
</html>
```

Db.py

```
import mysql.connector

# Establish database connection
mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword",
  database="yourdatabase"
)

# Get input username and password from user
username = input("Enter your username: ")
password = input("Enter your password: ")

# Check if user exists in database
mycursor = mydb.cursor()
mycursor.execute("SELECT * FROM users WHERE username=%s AND password=%s", (username,
```

```
password))
        result = mycursor.fetchone()

        if result:
          print("Login successful!")
        else:
          print("Invalid username or password.")
```

Output :

Login page:

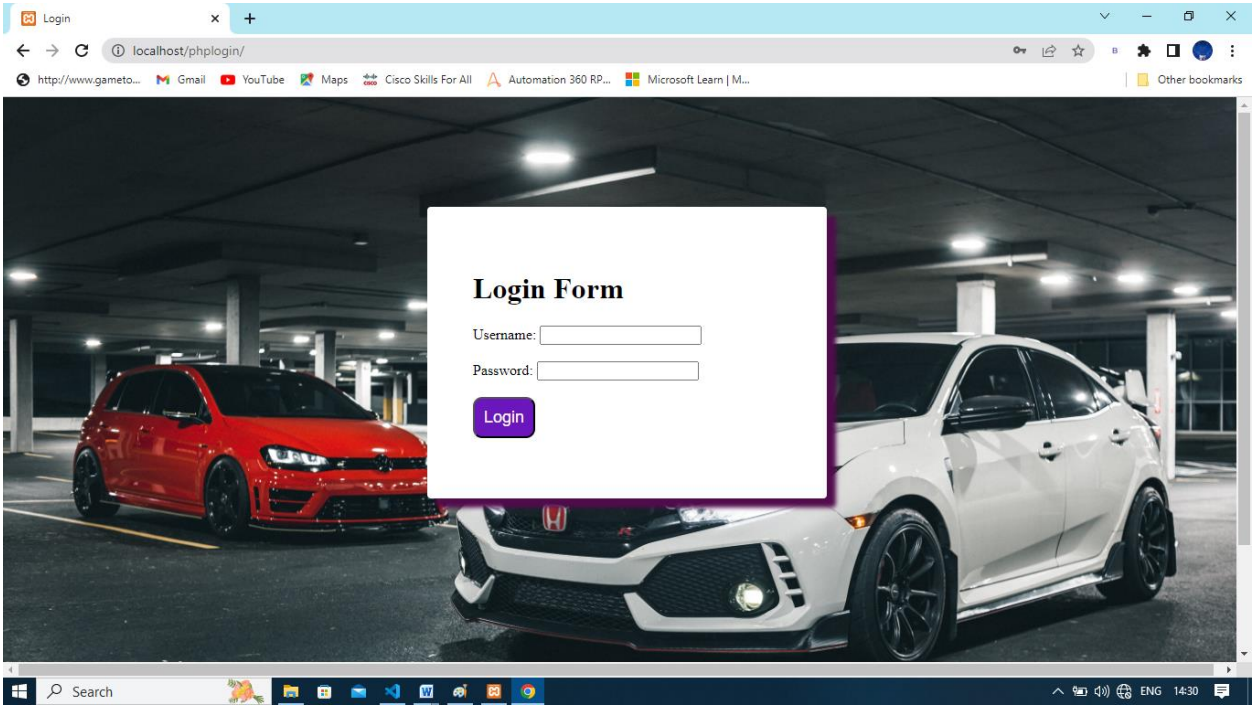


Figure 4. Login page

Index page:

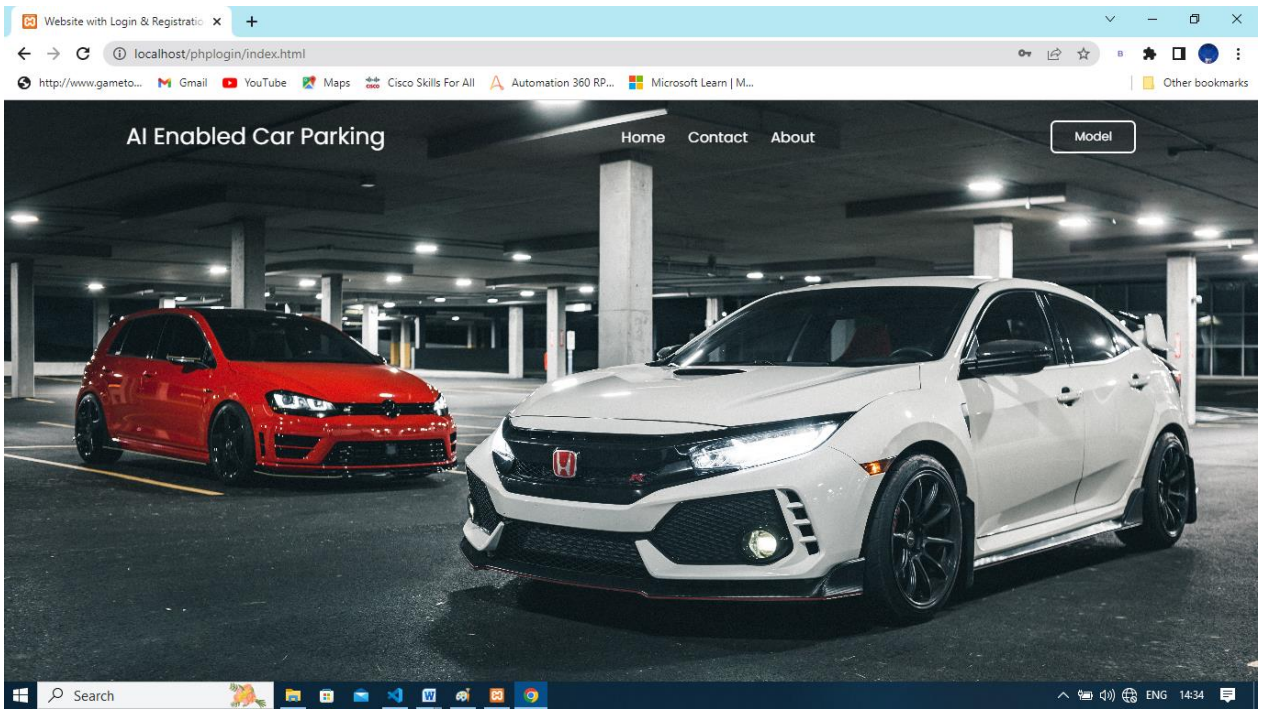


Figure 5. Index Page

Signup page:

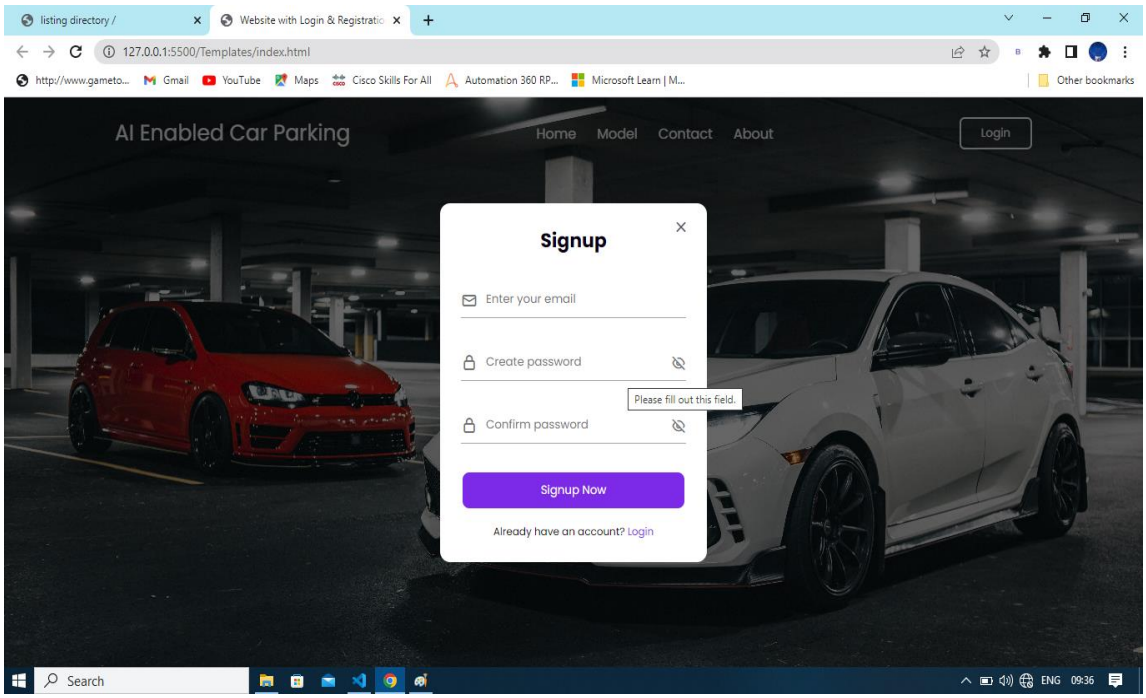


Figure 6. Signup Page

Model Page:

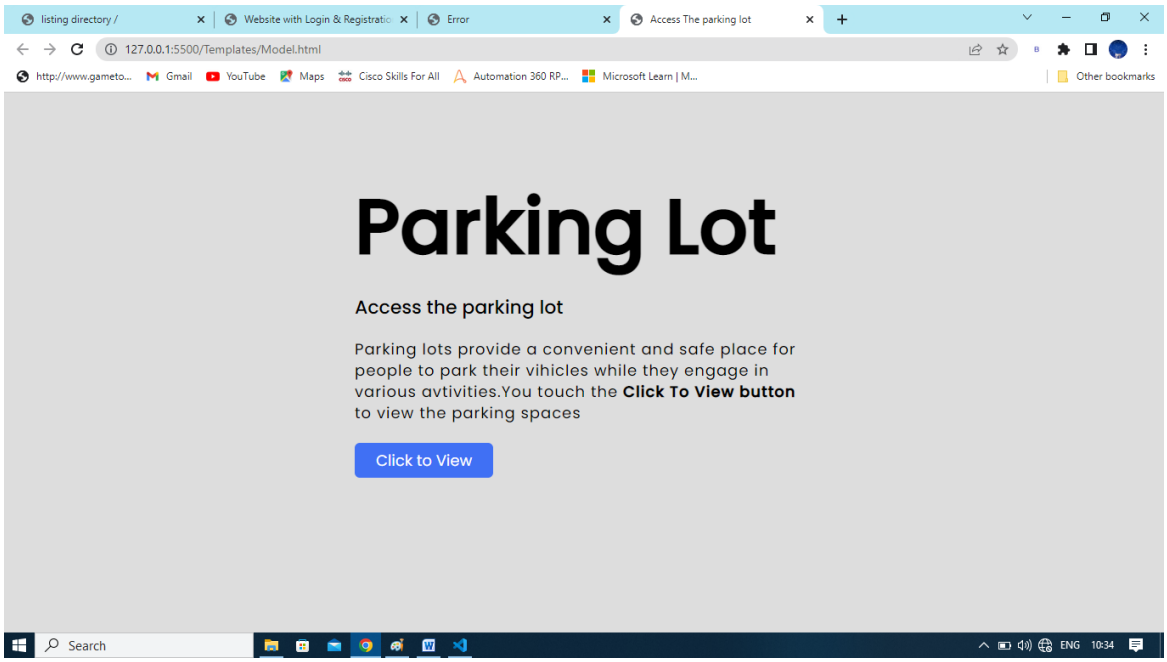


Figure 7. Model Page

Parking Slot Count :

Figure 8. Parking Slot Count

VII. 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.

VIII.FUTURE SCOPE

- Hook up a webcam to a snort Pi and have live parking monitoring at home
- alchemize parking lot video to have overview perspective(for clearer globules)
- It’s effective at resolving parking issues. In addition, it provides automatic billing, as well as eliminating traffic congestion. Utilising 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

IX.REFERENCES

1. Gaikwad, M.J., Asole, P.S. and Bitla, L.S., 2022, January. Effective Study of Machine Learning Algorithms for Heart Disease Prediction. In 2022 2nd International Conference on Power Electronics & IoT Applications in Renewable Energy and its Control (PARC) (pp. 1-6). IEEE.
2. A. Al-Smadi and M. Msallam, "Vehicle Auto Parking System," 2022 9th International Conference on Electrical and Electronics Engineering (ICEEE), 2022, pp. 108-111, doi: 10.1109/ ICEEE55327.2022.9772583.
3. B. K. Patil, A. Deshpande, S. Suryavanshi, R. Magdum and B. Manjunath, "Smart Parking System for Cars," 2018 International Conference on Recent Innovations in Electrical, Electronics & Communication Engineering (ICRIEECE), 2018, pp. 1118-1121, doi: 10.1109/ICRIEECE44171.2018.9008662
4. Y. Chunxia and A. Changsheng, "Research and Design of Automatic Parking System Based on Wechat Applet," 2021 4th International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE), 2021, pp. 997-1000,doi:10.1109/AEMCSE51986.2021.00203.
5. Keat, C.T.M.; Pradalier, C.; Laugier, C. Vehicle detection and car park mapping using laser scanner. In Proceedings ofthe IEEE/ RSJ International Conference on Intelligent Robots and Systems, Edmonton, AB, Canada, 2–6 August 2005; pp.2054–2