**CSE1901 - Technical Answers to Real World Problems (TARP)**

**Project Report**

**Parking Lot Automation**

*By*

19BCE1351 T Sri Siddhartha

19BAI1090 Arnab Karmakar

19BCE1774 Mahavir Lunkad

B. Tech Computer Science and Engineering

*Submitted to*

ASNATH VICTY PHAMILA Y

**School of Computer Science and Engineering**

****

*April 2022*

*­­­*

**DECLARATION**

I hereby declare that the report titled “PARKING LOT AUTOMATION**”** submitted by me to VIT Chennai is a record of bona-fide work undertaken by me under the supervision of ASNATH VICTY PHAMILA Y, School of Computer Science and Engineering, Vellore Institute of Technology, Chennai.

Signature of the Candidate

**T SRI SIDDHARTHA**

**Reg. No. 19BCE1351**

**ARNAB KARMAKAR**

**Reg. No. 19BAI1090**

**MAHAVIR LUNKAD**

**Reg. No. 19BCE1774**

**CERTIFICATE**

Certified that this project report entitled “PARKING LOT AUTOMATION**”** is a bonafide work of T Sri Siddhartha (19BCE1351), Arnab Karmakar (19BAI1090), Mahavir Lunkad (19BCE1774) and they carried out the Project work under my supervision and guidance for CSE1901 - Technical Answers to Real World Problems (TARP).

ASNATH VICTY PHAMILA Y

SCOPE, VIT Chennai

**ACKNOWLEDGEMENT**

Primarily, we would like to thank the almighty for all the blessings he showered over us to complete this project without any flaws.

The success and final outcome of this assignment required a lot of guidance and assistance from many people and we are extremely fortunate to have got this all along with the completion of our project. Whatever we have done is only due to such guidance and assistance by our faculty, ASNATH VICTY PHAMILA Y, to whom we are really thankful for giving us an opportunity to do this project.

Last but not the least, we are grateful to all our fellow classmates and our friends for the suggestions and support given to us throughout the completion of our project.

**T Sri Siddhartha**

**Reg. No. 19BCE1351**

**Arnab Karmakar**

**Reg. No. 19BAI1090**

**Mahavir Lunkad**

**Reg. No. 19BCE1774**

**ABSTRACT**

Due to the increase in the number of cars and other vehicles, it has become harder to find parking spots in populated and busy areas. Searching for an available parking space takes up a lot of potentially productive time, increases the number of cars circulating the roads by creating roadblocks and unnecessary traffic, and introduces challenges to the infrastructure.

Automatic parking lot systems help in reducing the land/space use for parking and maximizes the efficiency of its usage. Our project  aims to develop a prototype for an Automated parking lot system  that will increase the quality of service of parking lots through the integration of a smart system that assists motorists in finding vacant parking lots.

**CONTENTS**

Declaration i

Certificate ii

Acknowledgement iii

Abstract iv

1 Introduction 1

1.1 Problem Statement……………. . . . . . . . . . . . . . . . . . …….1

1.2 Objective and goal of the project. . . . . . . . . . . . . . . . . . .. . . 2

2 Literature Survey 6

3 Requirements Specification 7

3.1 Hardware Requirements . . . . . . . . . . . . . . . . . . . . . . . . . … 8

3.2 Software Requirements . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

4 System Design 9

5 Implementation of System 10

1. Results & Discussion 13

7 Conclusion and Future Work 16

8 References 17

Appendix <Sample code, snapshot etc.> 18

1. **Introduction**

**1.1 Problem Statement**

In recent times, the world has seen tremendous increase in the number of vehicle users. The availability of parking spaces, specifically in the Indian domain, has always seemed sparse. Very often, this results in compromising on pedestrian safety as the proprietors end up parking the car on unadvisable parking spaces (empty roads). To cut down on operation expenditure (manual labor cost) and increase public safety, while not sacrificing on the efficiency of the parking lot system, we have provided a low cost mobile application based IOT solution for vehicle parking lot automation.

**1.2 Objectives of Project**

This fully automated parking lot system that we have proposed makes use of the advantages of a mobile application's capabilities to generate unique QR codes, support self check-ins through number plate identification, and aid hassle free payment (UPI, netbanking). The project can be categorized into three distinct modules, namely; mobile app, number plate detection and parking space picker.

Initially, on opening the app, QR code is generated dynamically on the user's mobile application, right after the camera attached to the boom barrier detects the presence of a vehicle. Subsequently, the camera captures an image of the car's anterior and sends it to the machine learning backend of the model.

The user is then expected to show the generated QR code to the sensor for verification. The model then predicts the digital equivalent of the license plate number, which is later sent to the cloud-hosted database. The application keeps track of the user's credentials such as previous visits, transactions, license plate number and so on.

**2. Literature Survey**

Wendong Liao; Liang Xie; Jie Xi; Yan Bai; Teng Zhang; Yuqiong Wu et al, proposed a low-cost, high-efficiency, and easy-to-manage parking management system - a parking management system based on the Alibaba Cloud platform and machine learning [11].

1. Venkata SudhakaraA.V.Anoora Reddy K.Mounika M.V.Sai Kumar.Bharani et al, proposed a system that uses Image Processing techniques to identify the registration plate of a vehicle and provides autonomous door opening and closing operation whenever it detects a vehicle at entrance of a parking lot [12] .

Fan Jiang; Zhenglin Li; Minghao Tan; Qinyu Zhao et al, proposed a system that sends information of parking space availability to the computer through a Bluetooth module for remote monitoring by using the infrared digital obstacle avoidance sensor module and the RC522 card reading module records the times of swiping card and parking time and the serial screen display module displays parking space and exhaust gas concentration, and the real-time view of free parking space. A. Z. M. Tahmidul Kabir; Al Mamun Mizan; Plabon Kumar Saha; Md. Shajedul Hasan; Mohitosh Pramanik et al, implemented features like automated parking service, location tracking, parking management, real-time invoice generation, and payment system using IoT, Image processing, object detection, Firebase and GPS GSM modules.

In Mingjun Zhang’s paper, the vehicle license plate recognition system is broken down into two modules; the detection module and the recognition module. The detection module is further divided into two subcategories that involve two different methods to find the region of interest (ROI) and is judged by the Support vector machine (SVM) learning algorithm to ensure the reliability of the plate’s location. For the character recognition part, subsequent to pre-processing the images, acquired through detection, by normalization and conversion to grayscale, a modified version of the Le-Net5 network (pre-trained CNN model for simple numerals) is used. This model is said to produce predictions with a recognition rate of approximately 90%, even under extreme environmental conditions[6].

**3** **Requirements Specification**

3.1 **Hardware Requirements**

The system uses 3 camera modules in total out of which 2 of them are for edge computing and 1 for a live stream monitoring

3.2 **Software Requirements**

**Frontend:**

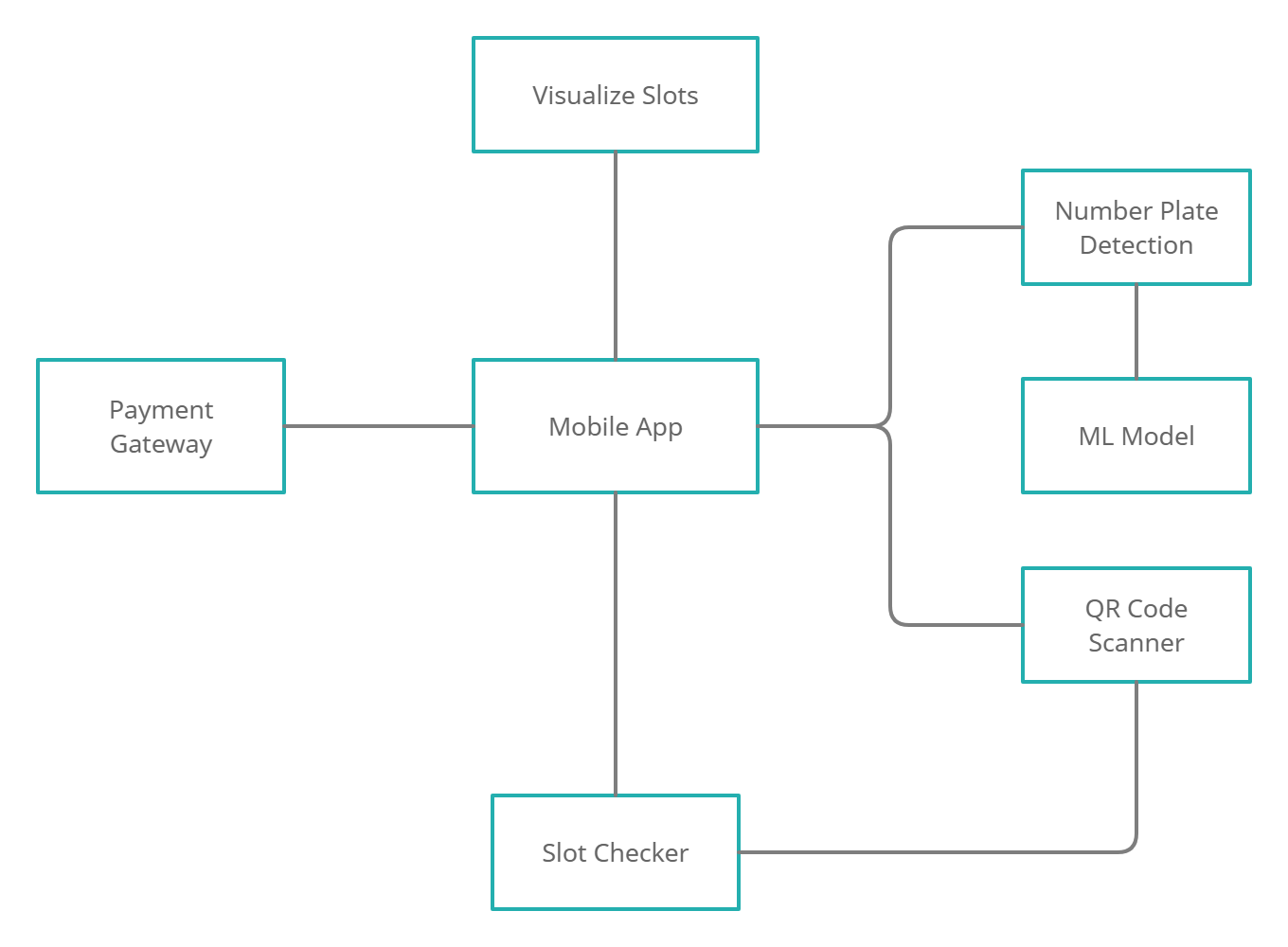
Flutter (Hybrid mobile app development framework)

Android Studio and ADB emulator

**Backend:**

Python (Pandas , OpenCV , Tensorflow , VideoGearApi )

**4 System Design**



1. **Implementation of System** 
   1. **PROPOSED WORK**

**Number plate recognition**

Image processing has single handedly been able to gratify all of the world’s image centric demands. The primal usage of the license plate detection module, with regards to the proposed parking lot system, is for the initial identification of the car and to enlist the proprietor’s credentials into the database. On capturing the QR information generated uniquely for each car entry, the camera attached to the boom barrier captures the live feed. A clear frame from the acquired input is taken to digitize the car’s number plate using the convolutional neural network model (CNN).

ALGORITHM

Assumptions: The indian vehicle license plate dataset contains all required types of alphanumeric data for optimal testing and detection of real-time inputs

Input: Unknown real-time image containing alphanumeric data

Output: Digital equivalent of the number plate

STEP 1:

Pre-processing of data by visualization, manipulation and conversion of the dataset from categorical form to vector (or binary) form.

STEP 1.1:

The color image is converted into a grayscale

STEP 1.2:

The pixel values are normalized using the built-in function.

STEP 2:

Creating the Convolutional Neural Network(CNN) model.

STEP 2.1: (Convolution Layer)

Generating feature maps by feature extraction of every individual image of the dataset.

STEP 2.2: (ReLu Layer)

All negative pixel values of the image are replaced by zeros to avoid summation of these values to zero.

STEP 2.3: (Pooling Layer)

The primary image matrix is broken down into 2x2 submatrices. Each submatrix is reduced to the value of its highest pixel number, in an effort to increase the computational speed.

STEP 2.4: (Fully Connected Layer)

The pooling layer matrices are pushed into a single list and the values which are higher are taken as the points of prediction for the respective image.

STEP 3:

The model is tested and evaluated after training it with a subset of the testing data. The result is obtained by understanding the difference in accuracy with each epoch

**Parking Space Picker**

To resolve the issue of the existing parking system we have come up with an IOT enabled parking space and allocation mechanism. The main objective of the Parking Space Picker module is to determine if the car is present in the respective parking slot or not. We have used Image Processing for the same using Python OpenCV.

STEP 1:

Firstly, we took a single frame and manually annotated the slots.

STEP 2:

The positions of these slots are then stored in a file.

STEP 3:

The main program iterated through the list of positions in the list.

STEP 4:

Then it isolates the region bounded by the coordinates to measure its pixel count in that region.

STEP 5:

Using this count value, we set a threshold value of 900 beyond which the slot is considered occupied.

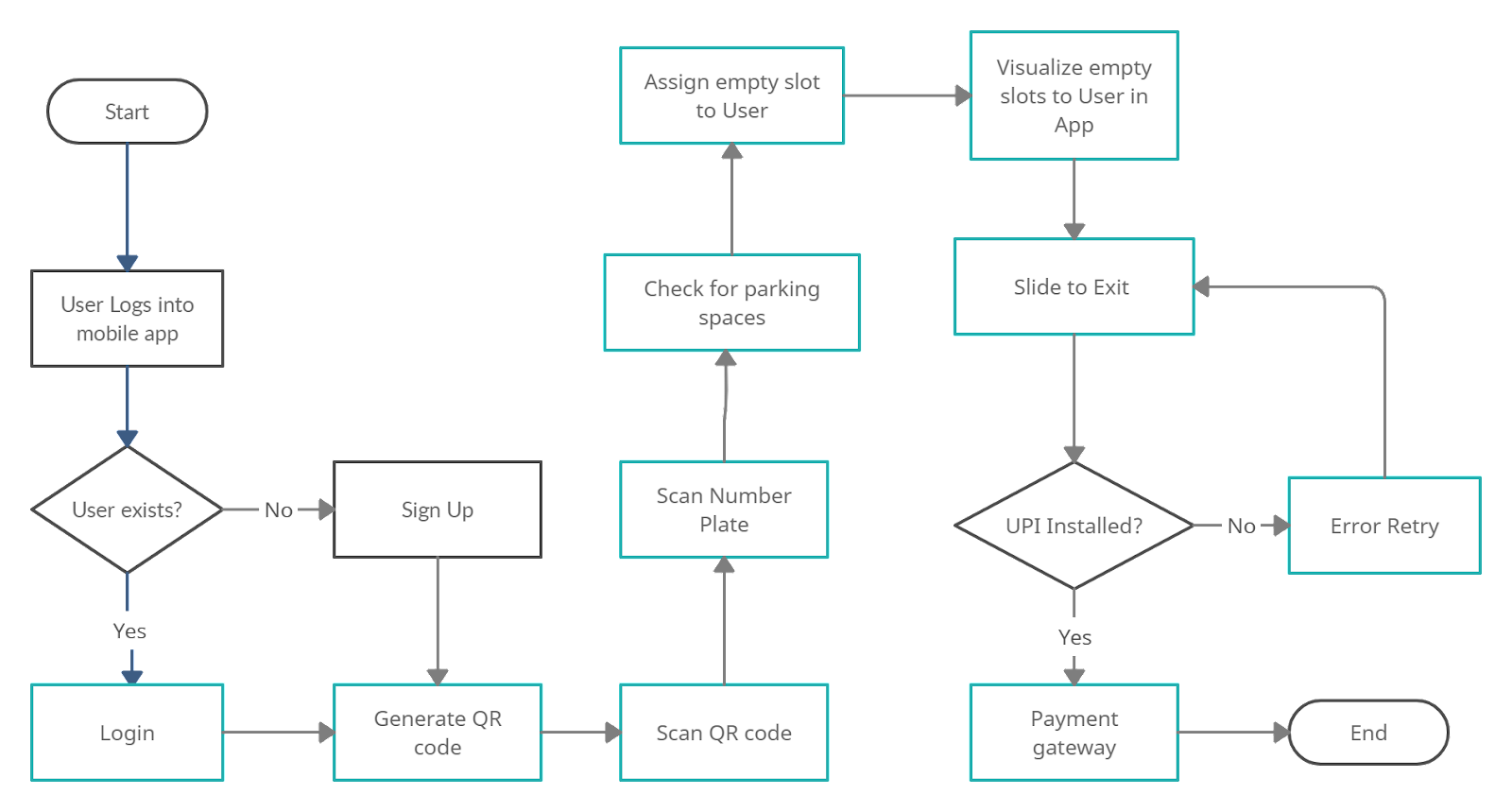
**Mobile Application**

This Parking lot system with a mobile app aims to completely automate the process of parking in offices, colleges and other recreational centers. The app is very simple to use and has a user-friendly UI/UX.

The core functionality includes generating a QR code that is to be scanned on entry and exit of a vehicle and a UI that visualizes and informs the user about the slot being allotted and all the other available slots so users don't need a person to guide them to a available slot The parking lot is monitored constantly to keep track of the filled and available slots using a camera positioned on the top

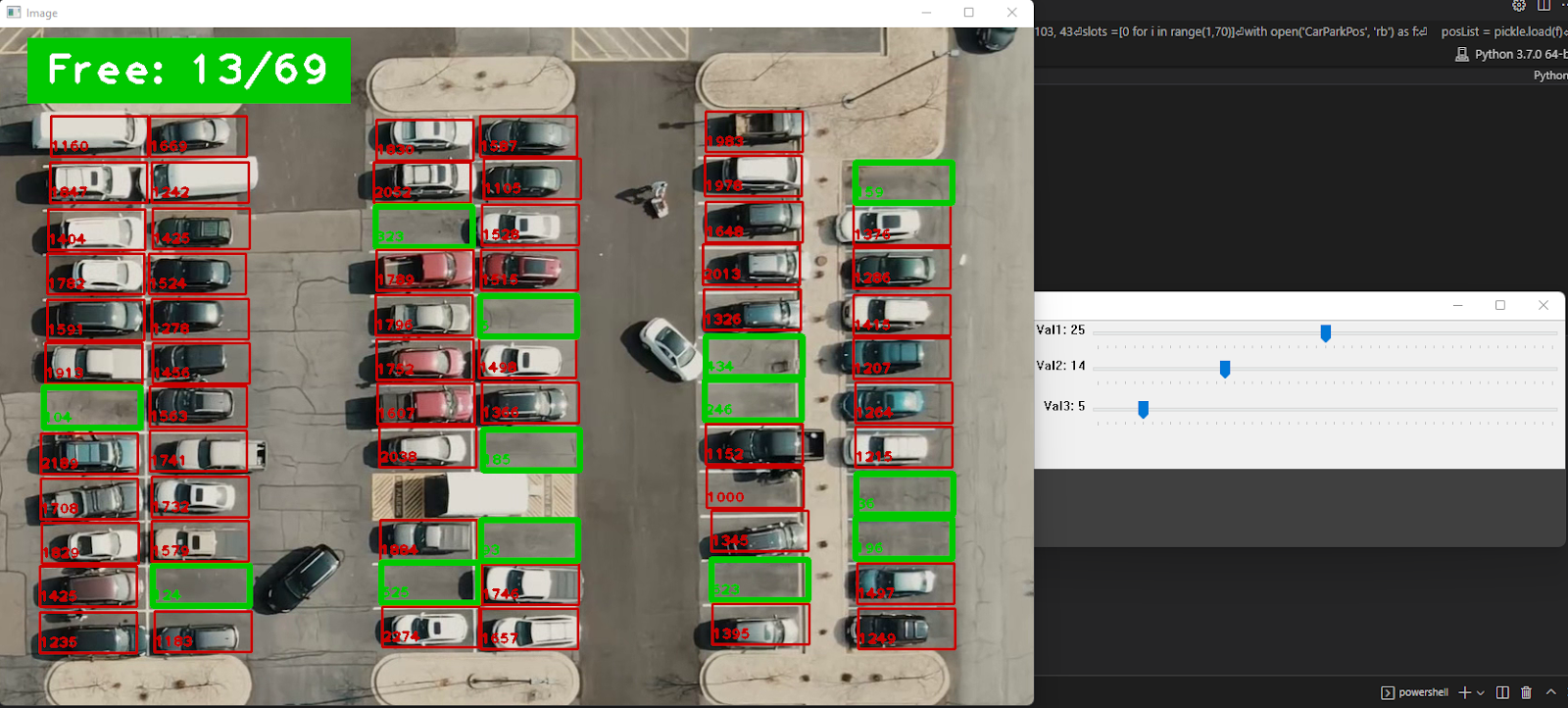
When the user exits the app automatically leads them to the payment page with minimal human intervention and making it completely online

This reduces unnecessary labor and infrastructure for managing the parking lot, dispensing tokens and receiving payments while sitting in a booth.

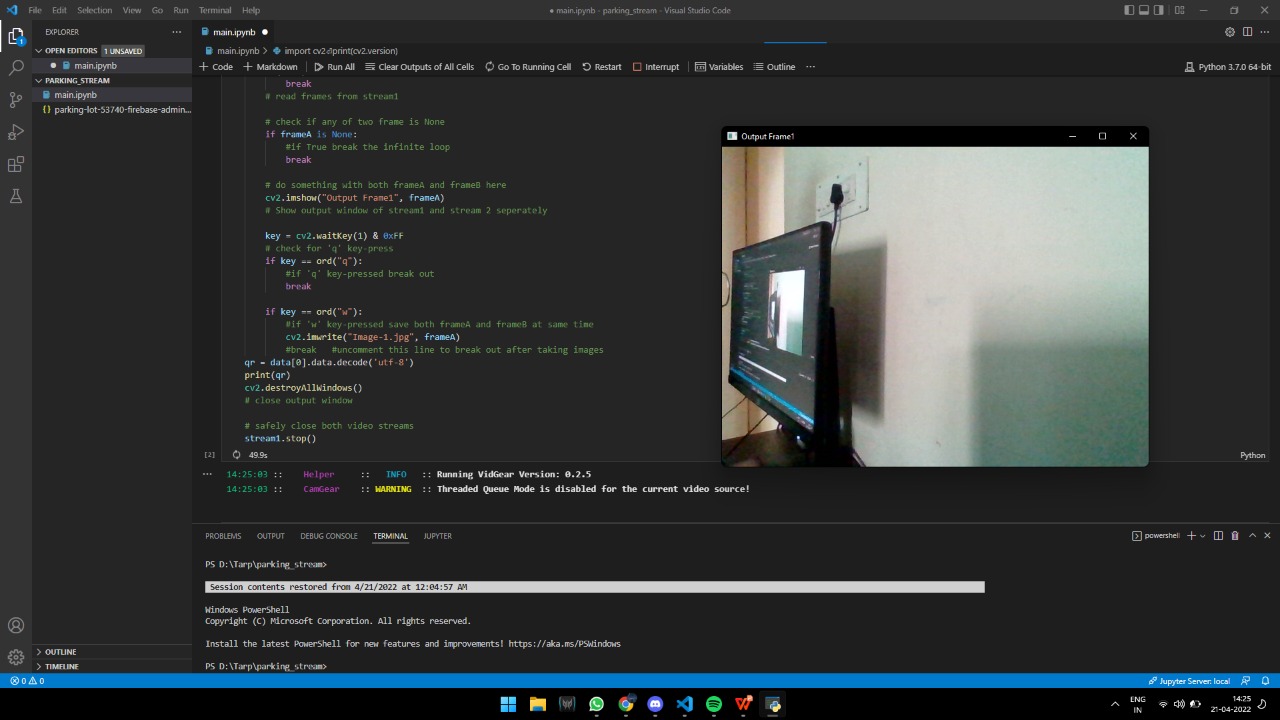
**Flow:**

1. **Results and Discussion**
   1. **Outputs and Screenshots**

**Parking Space Picker**

****

**QR Code Scanner**



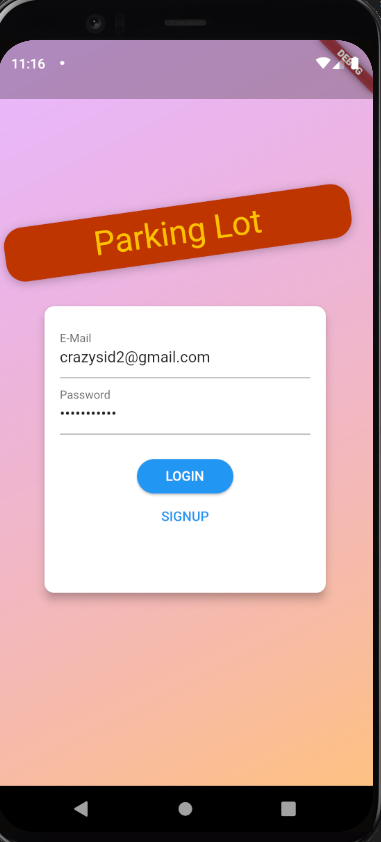
**Number Plate Detector**

****

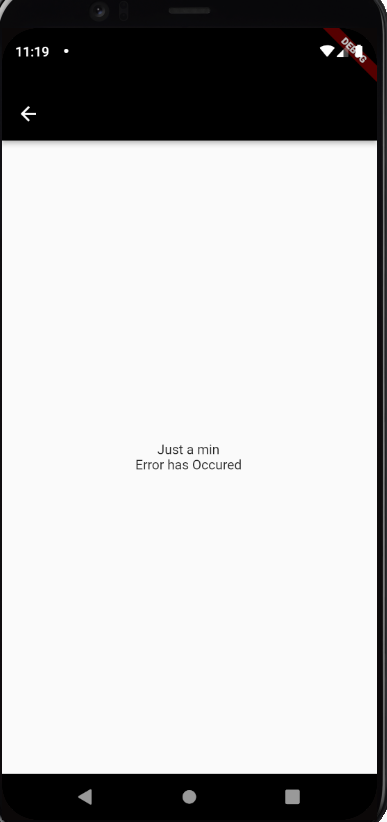
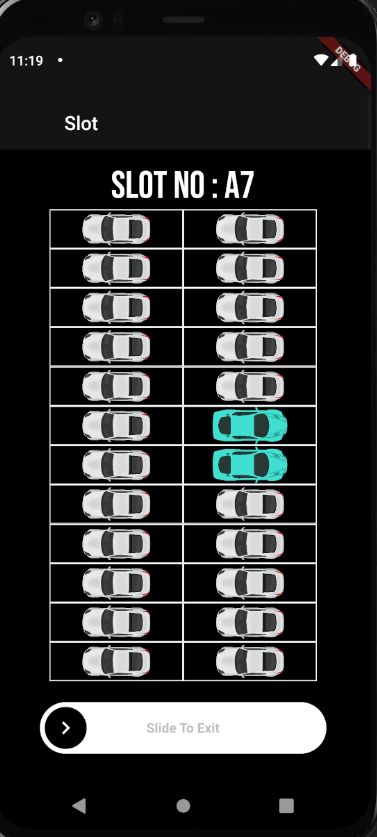
****

****

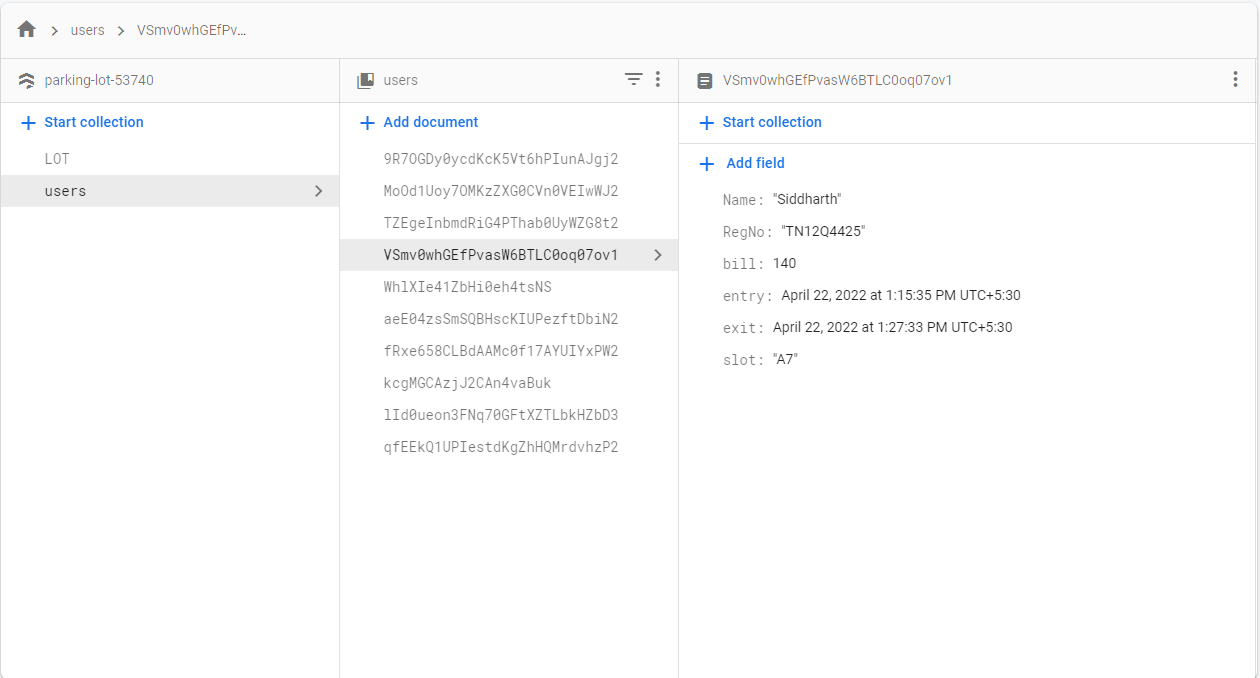
**App**



Parking Slot Visualizer Page Payment Gateway



**Firebase Bucket Console**



1. **Conclusion and Future Work**

Our Parking Lot Automation System was implemented using-  . With this automated parking lot system, a car with an authorized QR code can enter the parking lot without manually scanning the parking permit. This parking system also helps users view the availability of the parking space remotely. and thus helps the user in reducing the time needed to search for a parking space and also improves the parking lot utilization.

Our prototype system, Parking lot automation can be efficiently implemented into a real parking lot. This system also requires less maintenance and employee costs and the information gathered can be used in a distributed or centralized way to evaluate other useful metrics such as the parking duration, automatic billing as well as payment to the benefit of users and administrators. In the future, we plan to automate the process of finding each parking lot’s dimensions and the allocation of the bounding regions using machine learning. Our system currently uses a one time only manual annotation of the same, for a specific parking lot.

1. **REFERENCES**
2. https://ieeexplore.ieee.org/abstract/document/9498286
3. https://www.digiteum.com/iot-smart-parking-solutions/
4. K. M. Babu and M. V. Raghunadh, "Vehicle number plate detection and recognition using bounding box method," 2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT), 2016, pp. 106-110, doi: 10.1109/ICACCCT.2016.7831610.
5. M. Zhang, W. Yu, J. Su and W. Li, "Design of License Plate Recognition System Based on Machine Learning," 2019 IEEE 4th International Conference on Image, Vision and Computing (ICIVC), 2019, pp. 518-522, doi: 10.1109/ICIVC47709.2019.8981074.
6. S. Jain, R. Rathi and R. K. Chaurasiya, "Indian Vehicle Number-Plate Recognition using Single Shot Detection and OCR," 2021 IEEE India Council International Subsections Conference (INDISCON), 2021, pp. 1-5, doi: 10.1109/INDISCON53343.2021.9582216.
7. Ali, S., Li, J., Yan, P., Aslam, M. S., Shaukat, Z., & Azeem, M. (2020). An effective and improved CNN-ELM classifier for handwritten digits recognition and classification. Symmetry, 12(10), 1742. doi:http://dx.doi.org/10.3390/sym12101742
8. W. Liu, J. Wei and Q. Meng, "Comparisons on KNN, SVM, BP and the CNN for Handwritten Digit Recognition," 2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications( AEECA), 2020, pp. 587-590, doi: 10.1109/AEECA49918.2020.9213482.
9. International Journal of u- and e- Service, Science and Technology Vol.8, No.2 (2015), pp.251-262 http://dx.doi.org/10.14257/ijunesst.2015.8.2.25 ISSN: 2005-4246 IJUNESST Copyright ⓒ 2015 SERSC Understanding Smart and Automated Parking Technology
10. School of Mechanical Engineering, Xi’an University of Science and Technology, Xi’an 710054, China; a762323392@163.com (C.Z.); 20205016028@stu.xust.edu.cn (L.L.); 21205016036@stu.xust.edu.cn (X.Y.) \* Correspondence: 19205016023@stu.xust.edu.cn; Tel.: +86-183-9239-3130- Research on Automatic Parking System Strategy
11. D Annirudh;D Arun Kumar;Adabala Taraka Sai Raghava Kumar;K.R.M. Vijaya Chandrakala
12. 2021 IEEE Second International Conference on Control, Measurement and Instrumentation (CMI) - IoT based Intelligent Parking Management System
13. Intelligent parking lot control system based on Alibaba Cloud platform and machine learning Wendong Liao;Liang Xie;Jie Xi;Yan Bai;Teng Zhang;Yuqiong Wu - 2021 6th International Conference on Intelligent Computing and Signal Processing (ICSP)
14. Development of smart parking management system Author links open overlay panel M.Venkata SudhakaraA.V.Anoora ReddybK.MounikabM.V.Sai KumarbT.Bharanib

**APPENDIX**

**Backend**

**(Parking Space Counter - QR code scanner - Number Plate detector)**

# %%

import cv2

print(cv2.version)

# %% [markdown]

# ## Scan the user QR code

# %%

from vidgear.gears import VideoGear

from pyzbar.pyzbar import decode

import cv2

import time

# define and start the stream on first source ( For e.g #0 index device)

stream1 = VideoGear(source=0, logging=True).start()

detector = cv2.QRCodeDetector()

data = []

qr =""

# infinite loop

while True:

    frameA = stream1.read()

    data = decode(frameA)

    if(data):

        break

    # read frames from stream1

    # check if any of two frame is None

    if frameA is None:

        #if True break the infinite loop

        break

    # do something with both frameA and frameB here

    cv2.imshow("Output Frame1", frameA)

    # Show output window of stream1 and stream 2 seperately

    key = cv2.waitKey(1) & 0xFF

    # check for 'q' key-press

    if key == ord("q"):

        #if 'q' key-pressed break out

        break

    if key == ord("w"):

        #if 'w' key-pressed save both frameA and frameB at same time

        cv2.imwrite("Image-1.jpg", frameA)

        #break   #uncomment this line to break out after taking images

qr = data[0].data.decode('utf-8')

print(qr)

cv2.destroyAllWindows()

# close output window

# safely close both video streams

stream1.stop()

# %% [markdown]

# ## Check if slot is available for parking

# %%

import cv2

import pickle

import cvzone

import numpy as np

cap = cv2.VideoCapture('carPark.mp4')

width, height = 103, 43

slots =[0 for i in range(1,70)]

with open('CarParkPos', '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", 14, 50, empty)

cv2.createTrackbar("Val3", "Vals", 5, 50, empty)

def checkSpaces():

    spaces = 0

    for pos in posList:

        x, y = pos

        w, h = width, height

        ind = posList.index(pos)

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

        #print(imgThres[y, x])

        count = cv2.countNonZero(imgCrop)

        if count < 900:

            slots[ind] = 0

            color = (0, 200, 0)

            thic = 5

            spaces += 1

        else:

            slots[ind] = 1

            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)

    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

    if key == ord('q'):

        break

cv2.destroyAllWindows()

# %% [markdown]

# ## Intialize Firebase Server

# %%

import pyrebase

config = {

  "apiKey": "my-secret-api-key",

  "authDomain": "parking-lot-53740.firebaseapp.com",

  "databaseURL": "",

  "projectId": "parking-lot-53740",

  "storageBucket": "parking-lot-53740.appspot.com",

}

firebase = pyrebase.initialize\_app(config)

# %%

import firebase\_admin

from firebase\_admin import credentials ,firestore

cred = credentials.Certificate("./parking-lot-53740-firebase-adminsdk-lsefb-048bf58360.json")

firebase\_admin.initialize\_app(cred)

# %% [markdown]

# ## Check for empty slots and update to the server

# %%

firestore\_db = firestore.client()

snapshot = firestore\_db.collection('LOT').document('Lot1')

snapshot.update({

    'slots': slots

})

snep = firestore\_db.collection('LOT').get()

# %%

check = snep[0].to\_dict()['slots']

for i in range(len(check)):

    if(check[i] == 0):

        slotno = i+1

        break

if(slotno>=1 and slotno<=24):

    alph = 'A'

    num = slotno

elif(slotno>24 and slotno<=46):

    alph = 'B'

    num = slotno - 24

else:

    alph = 'C'

    num  = slotno - 46

fin = alph +str(num)

check[i] = 1

print(fin)

snapshot.update({

    'slots' : check

})

# %%

from datetime import datetime ,timezone

now = datetime.now(timezone.utc)

print(now)

# %%

user = firestore\_db.collection('users').document(qr)

def stream\_handler(message):

    print(message["event"]) # put

    print(message["path"]) # /-K7yGTTEp7O549EzTYtI

    print(message["data"]) # {'title': 'Pyrebase', "body": "etc..."}

user.update({

    'slot' : fin,

    'entry' : now,

    'RegNo' : "21BH2345AA"

})

# my\_stream = firestore\_db.collection('users').stream(stream\_handler)

# print(my\_stream.to\_dict())

# %%

user = firestore\_db.collection('users').document(qr).get()

u1 = user.to\_dict()

# %%

exit = datetime.now(timezone.utc)

diff = exit - u1['entry']

mins = int(round(diff.total\_seconds() / 60))

hours = mins//60

remain = mins % 60

if(remain>0):

    hours+=1

amount = 60 + (hours-1)\*40

user = firestore\_db.collection('users').document(qr)

user.update({

    'exit' : exit,

    'bill' : amount

})