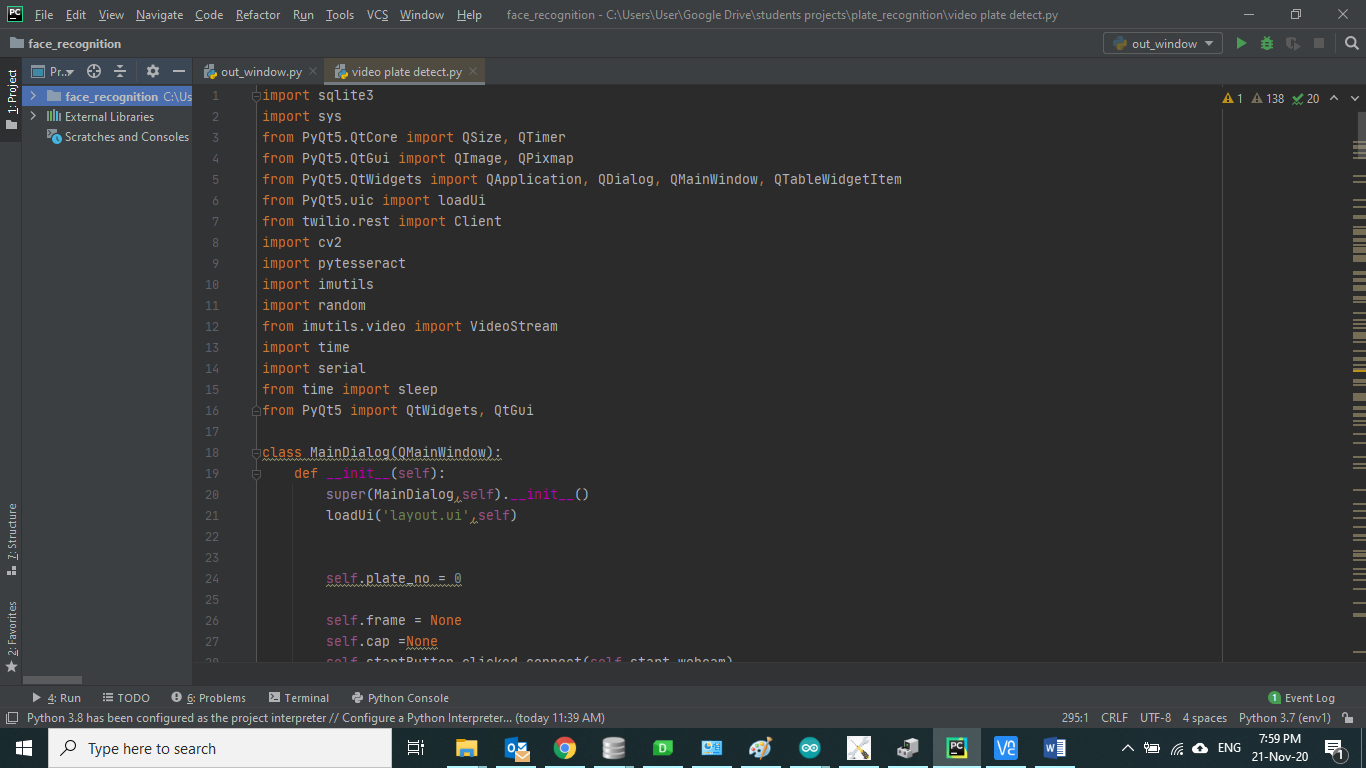
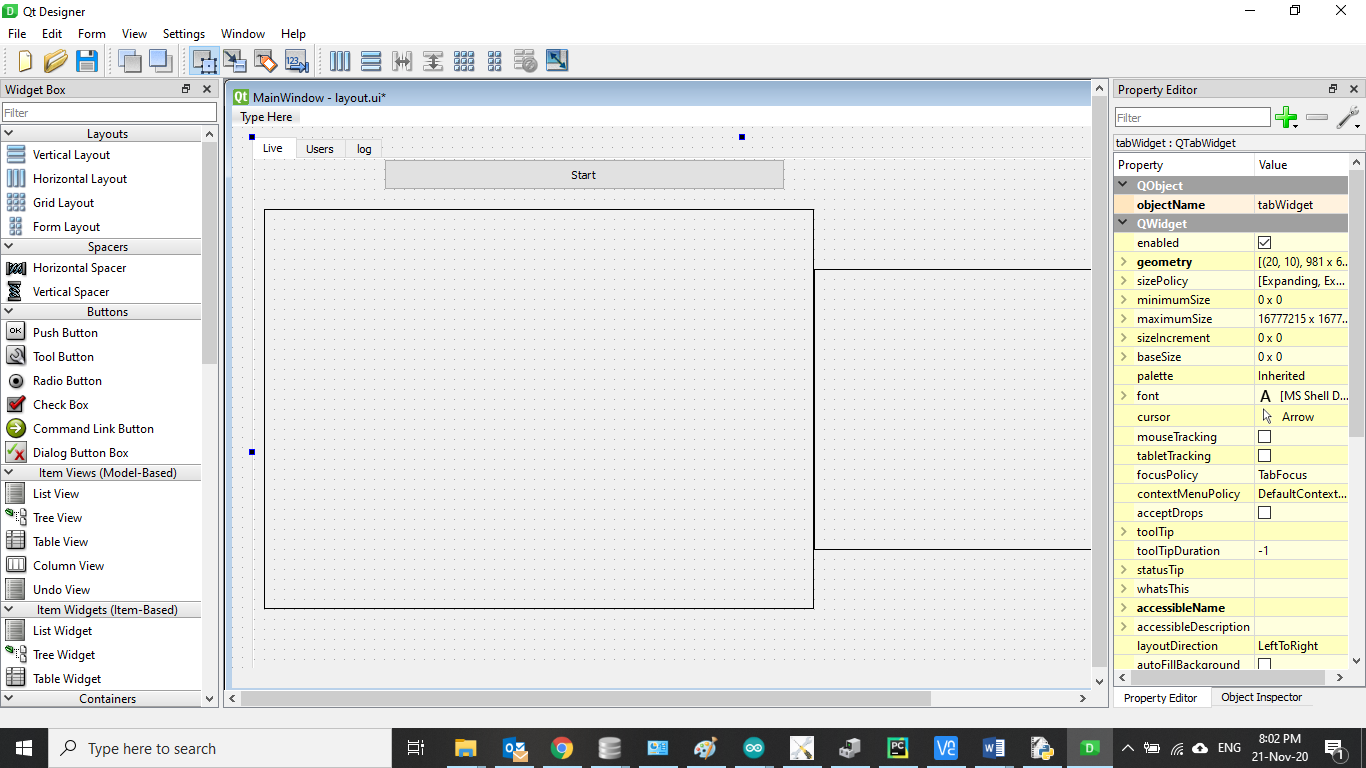
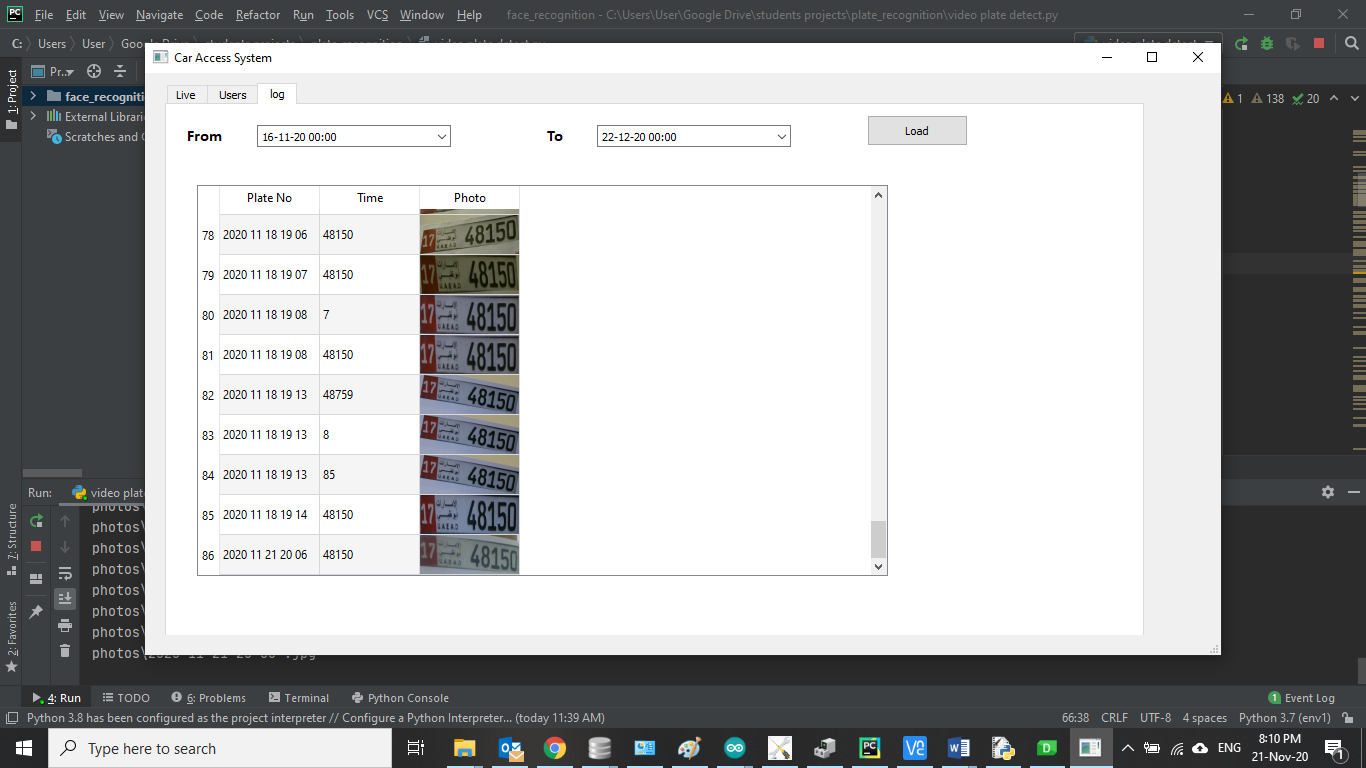
PYCHARM Integrated Development Environment

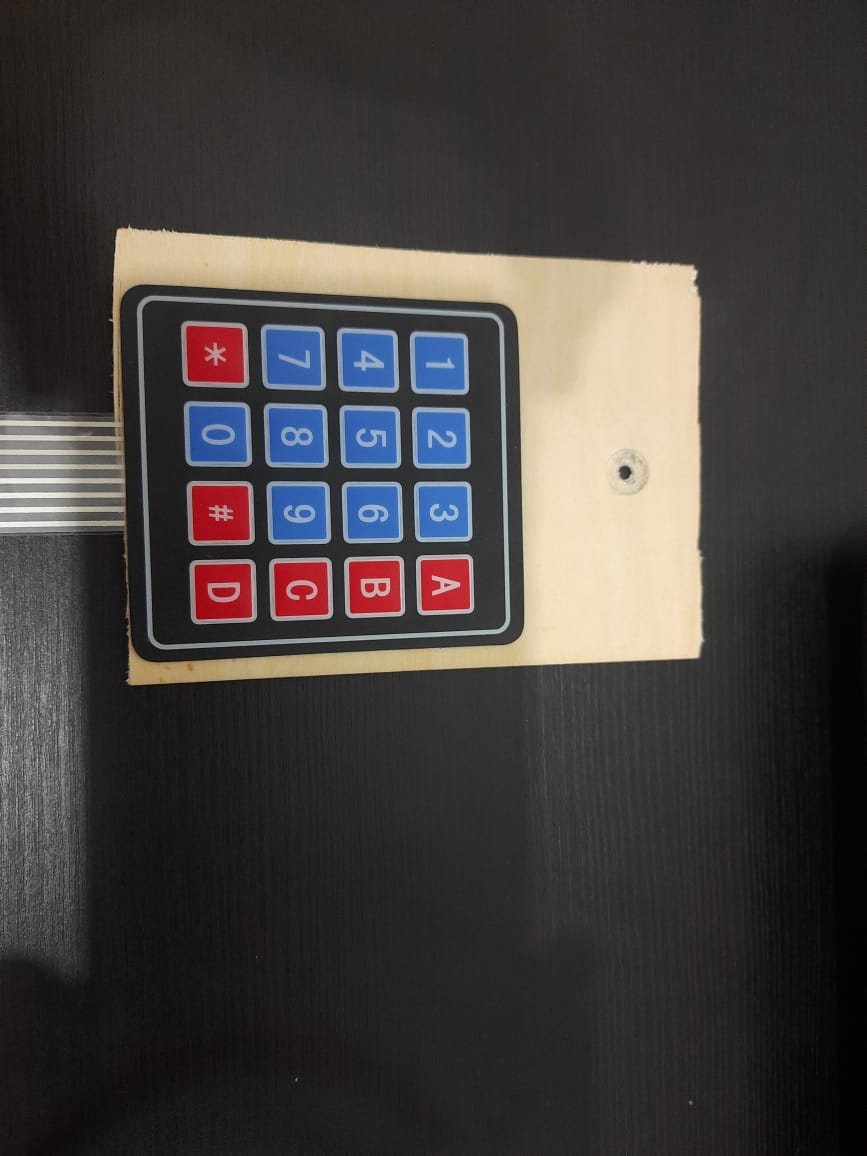
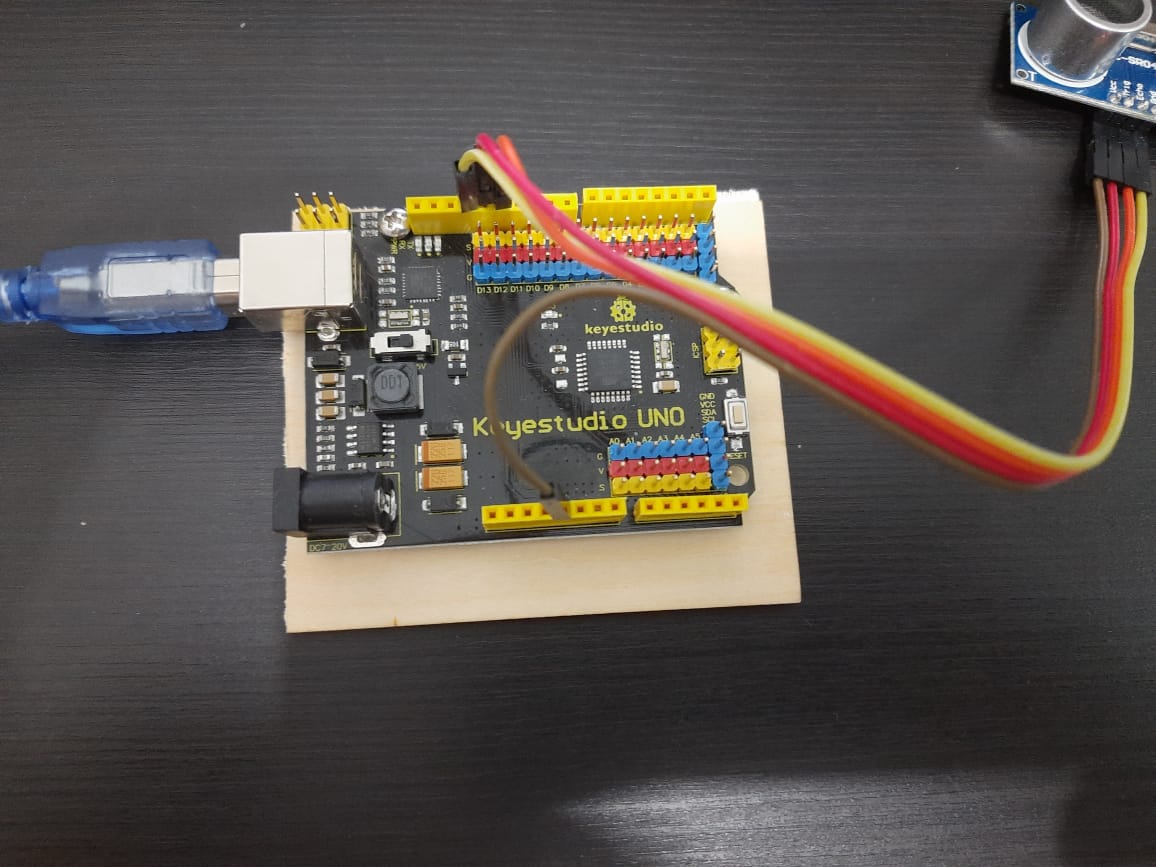


PYQT5 User interface Designer



Reports





The code

import sqlite3  
import sys  
from PyQt5.QtCore import QSize, QTimer  
from PyQt5.QtGui import QImage, QPixmap  
from PyQt5.QtWidgets import QApplication, QDialog, QMainWindow, QTableWidgetItem  
from PyQt5.uic import loadUi  
from twilio.rest import Client  
import cv2  
import pytesseract  
import imutils  
import random  
from imutils.video import VideoStream  
import time  
import serial  
from time import sleep  
from PyQt5 import QtWidgets, QtGui  
  
class MainDialog(QMainWindow):  
 def \_\_init\_\_(self):  
 super(MainDialog,self).\_\_init\_\_()  
 loadUi('layout.ui',self)  
  
  
 self.plate\_no = 0  
  
 self.frame = None  
 self.cap =None  
 self.startButton.clicked.connect(self.start\_webcam)  
 self.load.clicked.connect(self.loadLog)  
 # Initialize license Plate contour and x,y coordinates  
 self.contour\_with\_license\_plate = None  
 self.license\_plate = None  
 self.x = 1  
 self.y = 1  
 self.w = 1  
 self.h = 1  
 self.dev = serial.Serial('COM5', 9600, timeout=1) ## open serial port  
  
  
 def refreshTable(self):  
  
 db = sqlite3.connect('plateRecogDB.db')  
 cursor = db.cursor()  
  
 command = '''SELECT \* FROM users '''  
  
 result = cursor.execute(command)  
  
  
 ### to Fill table with result  
 self.table.setRowCount(0)  
 for row\_number, row\_data in enumerate(result):  
 self.table.insertRow(row\_number)  
 for column\_number, data in enumerate(row\_data) :  
 self.table.setItem(row\_number, column\_number, QTableWidgetItem(str(data)))  
  
  
  
 def start\_webcam(self): ## function of start button  
 #self.capture=cv2.VideoCapture(0) #0 =default #1,2,3 =Extra Webcam  
 # self.capture.set(cv2.CAP\_PROP\_FRAME\_WIDTH,480)  
 # self.capture.set(cv2.CAP\_PROP\_FRAME\_HEIGHT,640)  
 # self.vs = VideoStream(src=0).()  
 # self.frame = self.vs.read()  
 #self.frame = imutils.resize(self.frame, width=600)  
 self.cap = cv2.VideoCapture(1)  
 #self.cap = cv2.VideoCapture('rtsp://admin:INVENT123@192.168.1.64/1')  
 #self.cap = cv2.VideoCapture('rtsp://admin:INVENT123@192.168.1.64:554/videoMain')  
 self.timer = QTimer(self)  
 self.timer.timeout.connect(self.update\_frame)  
 self.timer.start(50)  
 self.timer4=QTimer(self)  
 def update\_frame(self):  
  
 #try :  
 ret, self.frame = self.cap.read()  
 #except:  
 #self.update\_frame()  
 if ret == True:  
  
 self.frame =cv2.resize(self.frame,(550,400))  
 img=self.frame #.copy()  
 #self.frame = imutils.resize(self.frame, width=600)  
 self.detected\_image = self.get\_no(img)  
  
 self.displayImage(self.detected\_image)  
  
  
  
 def get\_no(self, img):  
 # Convert to Grayscale Image  
 gray\_image = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
  
 #Canny Edge Detection  
 canny\_edge = cv2.Canny(gray\_image,30,200)  
  
 # Find contours based on Edges  
 contours, new = cv2.findContours(canny\_edge.copy(), cv2.RETR\_LIST, cv2.CHAIN\_APPROX\_SIMPLE)  
 contours=sorted(contours, key = cv2.contourArea, reverse = True)[:10]  
 #self.displyimage2(canny\_edge)  
  
 # Find the contour with 4 potential corners and creat ROI around it  
 for contour in contours:  
 # Find Perimeter of contour and it should be a closed contour  
 perimeter = cv2.arcLength(contour, True)  
 approx = cv2.approxPolyDP(contour, 0.018 \* perimeter, True)  
 if len(approx) == 4: #see whether it is a Rect  
 contour\_with\_license\_plate = approx  
 self.x, self.y, self.w, self.h = cv2.boundingRect(contour\_with\_license\_plate)  
 license\_plate = gray\_image[self.y: self.y + self.h, self.x: self.x + self.w]  
  
  
 # else:  
 # self.license\_plate = gray\_image  
  
  
 # Removing Noise from the detected image, before sending to Tesseract  
 license\_plate = cv2.bilateralFilter(license\_plate, 11, 17, 17)  
 #(thresh, license\_plate) = cv2.threshold(license\_plate, 150, 180, cv2.THRESH\_BINARY)  
  
 #Text Recognition  
 text = pytesseract.image\_to\_string(license\_plate)  
  
 #remove any wrong chracters and keep only numbers  
 tex=[]  
 for i in text:  
 if i in ['0','1','2','3','4','5','6','7','8','9'] :  
 tex.append(i)  
 text= ''.join([str(elem) for elem in tex])  
 if len(text) != 0:  
 print("License Plate :", text)  
 self.plate\_no = int(text)  
 self.displyimage2(canny\_edge)  
 self.insertlog(text,img[self.y: self.y + self.h, self.x: self.x + self.w])  
  
 self.checkandSendSMS(text)  
 # #Draw License Plate and write the Text  
 img = cv2.rectangle(img, (self.x,self.y), (self.x+self.w,self.y+self.h), (0,0,255), 3)  
 #img = cv2.putText(img, text, (self.x-100,self.y-50), cv2.FONT\_HERSHEY\_SIMPLEX, 3, (0,255,0), 6, cv2.LINE\_AA)  
 time.sleep(1)  
  
 #cv2.imshow("Frame",self.frame)  
 return img  
 def loadLog(self):  
 format = "yyyy MM dd hh mm";  
 print(self.fromTime.dateTime().toString(format))  
 db = sqlite3.connect('plateRecogDB.db')  
 cursor = db  
 command = '''SELECT \* FROM log WHERE time BETWEEN ? AND ? '''  
 row= (self.fromTime.dateTime().toString(format),self.toTime.dateTime().toString(format))  
 result = cursor.execute(command,row)  
  
 ### to Fill table with result  
 self.table.setRowCount(0)  
 for row\_number, row\_data in enumerate(result):  
 self.table.insertRow(row\_number)  
 for column\_number, data in enumerate(row\_data):  
 if(column\_number==2):  
 item=self.getImageLable(data)  
 self.table.setCellWidget(row\_number,column\_number,item)  
 else :  
 self.table.setItem(row\_number, column\_number, QTableWidgetItem(str(data)))  
 self.table.verticalHeader().setDefaultSectionSize(40)  
  
 def getImageLable(self,imName):  
 imageLabel =QtWidgets.QLabel(self.log)  
 imageLabel.setText("aa")  
 imageLabel.setScaledContents(True)  
 #pixmap = QtGui.QPixmap("2020 11 16 17 26 04.jpg")  
 imagename =str("photos\{0}.jpg".format(str(imName)))  
 print(imagename)  
  
 pixmap = QtGui.QPixmap(str("photos\{0}.jpg".format(imName)))  
  
 #pixmap.loadFromData(image,'jpg')  
 imageLabel.setPixmap(pixmap)  
 return imageLabel  
  
  
 def insertlog(self, text,img):  
 time\_ = str(time.strftime("%Y %m %d %H %M "))  
 print(time\_)  
 filename = time\_.replace(':', ' ')  
 print(filename)  
 cv2.imwrite("photos\{0}.jpg".format(filename), img)  
 db = sqlite3.connect('plateRecogDB.db')  
 cursor = db.cursor()  
 row = (time\_, str(text), time\_)  
 command = '''INSERT INTO log (time ,plate,photo) VALUES (?,?,?)'''  
 cursor.execute(command, row)  
 db.commit()  
  
 def checkandSendSMS(self,text):  
 db = sqlite3.connect('plateRecogDB.db')  
 cursor = db.cursor()  
 command = '''SELECT tel FROM users WHERE plate\_no=? '''  
 result = cursor.execute(command,[text]).fetchone()  
 if result:  
 tel=result[0]  
 # Generate rendom OTP  
 otp = random.randint(1000, 9999)  
 print(otp)  
 # send SMS  
 # Your Account SID from twilio.com/console  
 account\_sid = "ACcb66f8276f5ba0ece809e9a2552fe845"  
 # Your Auth Token from twilio.com/console  
 auth\_token = "8b7a4f808fd9aa2f4a554cae4755b9a6"  
  
 client = Client(account\_sid, auth\_token)  
  
 message = client.messages.create(  
 to=str('+'+str(tel)),  
 from\_="+12314621944",  
 body="use this OTP to Open the Gate " + str(otp))  
 self.dev.flushInput()  
 self.dev.flushOutput()  
 self.getOtpFromKeypad(otp)  
  
 def getOtpFromKeypad(self,otp):  
 data = ""  
 line = ''  
 while line != 'A':  
 #time.sleep(.001)  
 line = self.dev.readline().decode().strip()  
 if line != 'A':  
 if line != '':  
 data=data + line  
 print(data)  
 if data == str(otp):  
 print('open door')  
 self.dev.write(b'1')  
 #self.timer4.singleShot(2000,self.checkSensor)  
 self.timer4.timeout.connect(self.checkSensor) # Connect timeout to the output function  
 self.timer4.start(3000)  
  
 else :  
 print('wrong OTP')  
  
 def checkSensor(self):  
 self.dev.flushInput()  
 self.dev.flushOutput()  
 self.dev.write(b'2')  
 time.sleep(3)  
  
 #print(self.dev.readline())  
 distance = int(self.dev.readline().decode().strip())  
 if distance < 30 :  
 print ('car Reached Parking')  
 self.updateUserStatus('in')  
 else :  
 print('car did not reach the parking')  
 self.updateUserStatus('out')  
 self.timer4.stop()  
  
 def updateUserStatus(self,str):  
  
  
 db = sqlite3.connect('plateRecogDB.db')  
 cursor = db  
 command = '''UPDATE users SET status =? WHERE plate\_no =?'''  
 row = (str ,self.plate\_no)  
 cursor.execute(command, row)  
 db.commit()  
  
  
 def displayImage(self, img ):  
 qformat = QImage.Format\_Indexed8  
  
 if len(img.shape) == 3: # rows[0],cols[1],channels[2]  
 if (img.shape[2]) == 4:  
 qformat = QImage.Format\_RGBA8888  
 else:  
 qformat = QImage.Format\_RGB888  
 img = QImage(img, img.shape[1], img.shape[0],img.strides[0], qformat)  
 # BGR > RGB  
 img = img.rgbSwapped()  
 self.imgLable.setPixmap(QPixmap.fromImage(img))  
 self.imgLable.setScaledContents(True)  
  
 def displyimage2(self,img):  
## disply image2  
 qformat = QImage.Format\_Indexed8  
  
 if len(img.shape) == 3: # rows[0],cols[1],channels[2]  
 if (img.shape[2]) == 4:  
 qformat = QImage.Format\_RGBA8888  
 else:  
 qformat = QImage.Format\_RGB888  
 img = QImage(img, img.shape[1], img.shape[0], img.strides[0], qformat)  
 # BGR > RGB  
 img = img.rgbSwapped()  
 self.imgLable2.setPixmap(QPixmap.fromImage(img))  
 self.imgLable2.setScaledContents(True)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app = QApplication(sys.argv)  
 window = MainDialog()  
 window.setWindowTitle('Car Access System')  
 window.show()  
 sys.exit(app.exec\_())