NEW HORIZON COLLEGE OF ENGINEERING
Autonomous College, Affiliated to VTU | Approved by AICTE New Delhi & UGC
Accredited by NAAC with 'A' Grade & Accredited by NBA

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

A MINI PROJECT REPORT ON

"QUICK-SCAN"

Submitted in the partial fulfillment of the requirements in the 3rd semester of

BACHELOR OF ENGINEERING
IN

INFORMATION SCIENCE AND ENGINEERING

FOR

COURSE NAME: MINI PROJECT

COURSE CODE: 20ISE391A

By

CHETHAN A S - 1NH20IS035

HEMSAGAR – 1NH20IS062

Under the guidance of

Mrs. SWATHI

NEW HORIZON
COLLEGE OF ENGINEERING
Autonomous College, Affiliated to VTU | Approved by AICTE New Delhi & UGC
Accredited by NAAC with 'A' Grade & Accredited by NBA

CERTIFICATE

Certified that the project work entitled "QUICK-SCAN" carried out by Mr. CHETHAN A S and Mr. HEMSAGAR N M bearing USN 1NH20IS035 and USN 1NH20IS062 respectively bonifide students of III semester in partial fulfillment for the award of Bachelor of Engineering, an autonomous institute affiliated to the Visvesvaraya Technological University, Belagavi during the year 2021-2022. It is certified that all corrections / suggestions for Internal Assessment have been incorporated. The project report has been approved as it satisfies the academic requirements in respect of Mini Project work prescribed for the said Degree**.**

Name & Signature of Guide	**Name & Signature of HOD**
Mrs. Swathi	Dr. Anandhi R J

Examiners:

	Name	**Signature**
1.	**……………………………………**	**………………………**
2.	**……………………………………**	**……………………….**

ACKNOWLEDGEMENT

Any project is a task of great enormity and it cannot be accomplished by an individual without support and guidance. I am grateful to a number of individuals whose professional guidance and encouragement has made this project completion a reality.

We have a great leasure in expressing our deep sense of gratitude to the beloved Chairman **Dr. Mohan Manghnani** for having provided us with a great infrastructure and well-furnished labs.

We take this opportunity to express our profound gratitude to the Principal **Dr.Manjunatha** for his constant support and management.

We are grateful to **Dr. R J Anandhi**, Professor and Head of Department of ISE, New Horizon College of Engineering, Bengaluru for her strong enforcement on perfection and quality during the course of our mini project work.

We would like to express our thanks to the guide **Mr. Gangadhar Immadi**, Senior Assistant Professor, Department of ISE, New Horizon College of Engineering, Bengaluru who has always guided us in detailed technical aspects throughout our mini project.

We would like to mention special thanks to all the Teaching and Non-Teaching staff members of Information Science and Engineering Department, New Horizon College of Engineering, Bengaluru for their invaluable support and guidance.

CHETHAN A S 1NH20IS035

HEMSAGAR N M 1NH20IS062

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Quick-Scan is an application system used to scan QR present in the tickets and inspect the ticket's details if its present in the database file or not and valid and invalid tickets can be easily identified. The scanned and decoded data will be stored in the respective transportation companies database.

Embezzlement and people using counterfeit tickets in public transit are the biggest loop holes in the industry. In the mass transport transit like trains the ticket inspector can not inspect the tickets of every passenger in the transit and it is a hassle and non productive work where many passengers will get away from not inspectors easily, by using Quick-scan system the tickets inspection is done quickly and efficiently.

Data stored in the data base file will help the public transportation companies to better understanding the usage of their service by the public on day to day basis.

Comments:
Name & Sign of Committee Members:
1.
2.
3.

Approval Status		
Approved	Pending	Rejected

CHAPTER 1

1.1 INTRODUCTION

Our “QUICK-SCAN” is a system application where QR-code or Barcode can be scanned to fetch the data present in the QR/Barcode. The data scanned from the QR/Barcode can be stored in a ‘database’ or it can be searched in database to check if the data required is present in the corresponding database.

“QUICK-SCAN” system can be used in wide range of industries where there is a need of collection of data in an efficient and convenient way. This system can be used as an application for data analytics and data research where it deals with mass data handling.

Our program is used as an application to store the data of the tickets and inspect the tickets data rapidly by scanning the QR or Barcode present in tickets in a very efficient manner which saves time, data collection about transit customer like fair-price, route and other transit related data can be easily collected into a database while inspecting the data of the tickets.

1.2 Motivation of the project

Inspecting tickets is done manually in the transportation sector and it is very time-consuming process and there no current system to store the data present in the inspected tickets which is much need to a transportation company to keep track on its customers using their services.

Counterfeit tickets and disloyal employee of the transportation company who can do an act of embezzlement can be tackled by using our system which can inspect every ticket detail in the transit and able to register the scanned ticket details.

Corruption in the inspecting department can be brought to an end where the inspector provides fake details about inspecting the tickets of the passenger of the transit our system does not allow such loop holes to take place by using our Quick-scan system.

1.3 Problem Definition

In present day transportation industry embezzlement by the conductor and people using counterfeit or older tickets to travel in a transit is a huge loop-hole in the transportation service and due to this problem, there is huge money being not making into the right table of the respective company where the money should reach to their accounts.

Inspection of tickets is done manually in every transit and it is very hassle and time-consuming process, during hectic situation the ticket inspector cannot inspect all the tickets of the passengers and the passengers with the fake or counterfeit tickets can easily get away from the ticket inspector. There is no practice of collecting the data of inspected tickets so there is no clear picture between the company and the man power i.e inspectors inspecting the tickets.

In huge transit like trains where mass number of passengers travel on daily basis inspecting the tickets is a hectic process and many passengers travel without tickets and fake counterfeit tickets and there will be no stored evidence of inspected tickets by the ticket inspector, by using our system by scanning the QR-code or barcode the details the data of ticket details will be registered or can be inspected to check if the details is present in the database of the transportation company.

CHAPTER 2
LITERATURE SURVEY

2.1 Existing System

As of now there is only manual ticket inspection done by the public transportation services, A ticket inspector is a person whose task that would be to scrutinize the tickets of public transportation commuters.

Manual ticket inspection is a time-consuming process and hassle process and there is technology incorporated to ticket inspection system process by any public transportation companies.

❖ Limitations

- Manual ticket inspection is a hassle and a long and arduous process
- It requires a greater number of inspectors to inspect the tickets of mass number of passengers
- Lack of incorporation of the technology to the system

2.2 Proposed System

‘Quick-Scan’ is the name of our proposed system which concerns about inspecting the details of the tickets on inspection and also storing the details data of the tickets for inspection by storing it in the database for the respective transportation company.

The data details of the tickets stored in the database file can used for surveying and monitor the public using their transportation service. Data is money in present day world and by looking into the data respective public transportation company can get an outlook about how their travel business is doing on day-to-day basis.

Quick-scan uses the QR-code/Barcode scanner which detects the QR/Barcode present on the tickets and decodes it into required data form and stores the decoded data into an database or we can use the scanner to inspect the ticket details and the system can detect if the details of the ticket is present in the company’s database or not and provides specific message as an result.

Using a scanner to inspect the tickets is a one of the efficient methods to quickly inspect the tickets and register the ticket’s data into a file and the ticket inspector job is made more productive than inspecting the tickets manually.

2.3 Objectives of the Proposed System

- To stop the loop-holes present in the public transportation service.
- Creating a small database to collect the data of each and every ticket which has been generated in the transit and inspected by the inspector.
- Inspecting the tickets for the data and to check if its details is registered in the data base using the QR code scanner.
- Learn how to implement Machine Learning technology in our system.
- To make ticket inspection more effective and productive.

CHAPTER 3

SYSTEM REQUIREMENTS SPECIFICATION

3.1 Hardware Requirements

The following are needed to efficiently use the application.

Processor	-	Intel Core i3 and above
Speed	-	2.5 GHz
RAM	-	8 GB (min)
Hard Disk	-	50 GB
Camera	-	webcam(0.9MP,16:9(1280*720))

3.2 Software Requirements

Software requirements define software resource fundamentals that need to be installed on a workstation to provide optimum working of a software. The following are required for optimal development and usage of the application.

Operating System	-	Windows 7 and above
Programming Language	-	Python 3.9/3.10
Compiler	-	PyCharm, IDLE(Python)

CHAPTER 4

SYSTEM DESIGN

4.1.1 System Architecture

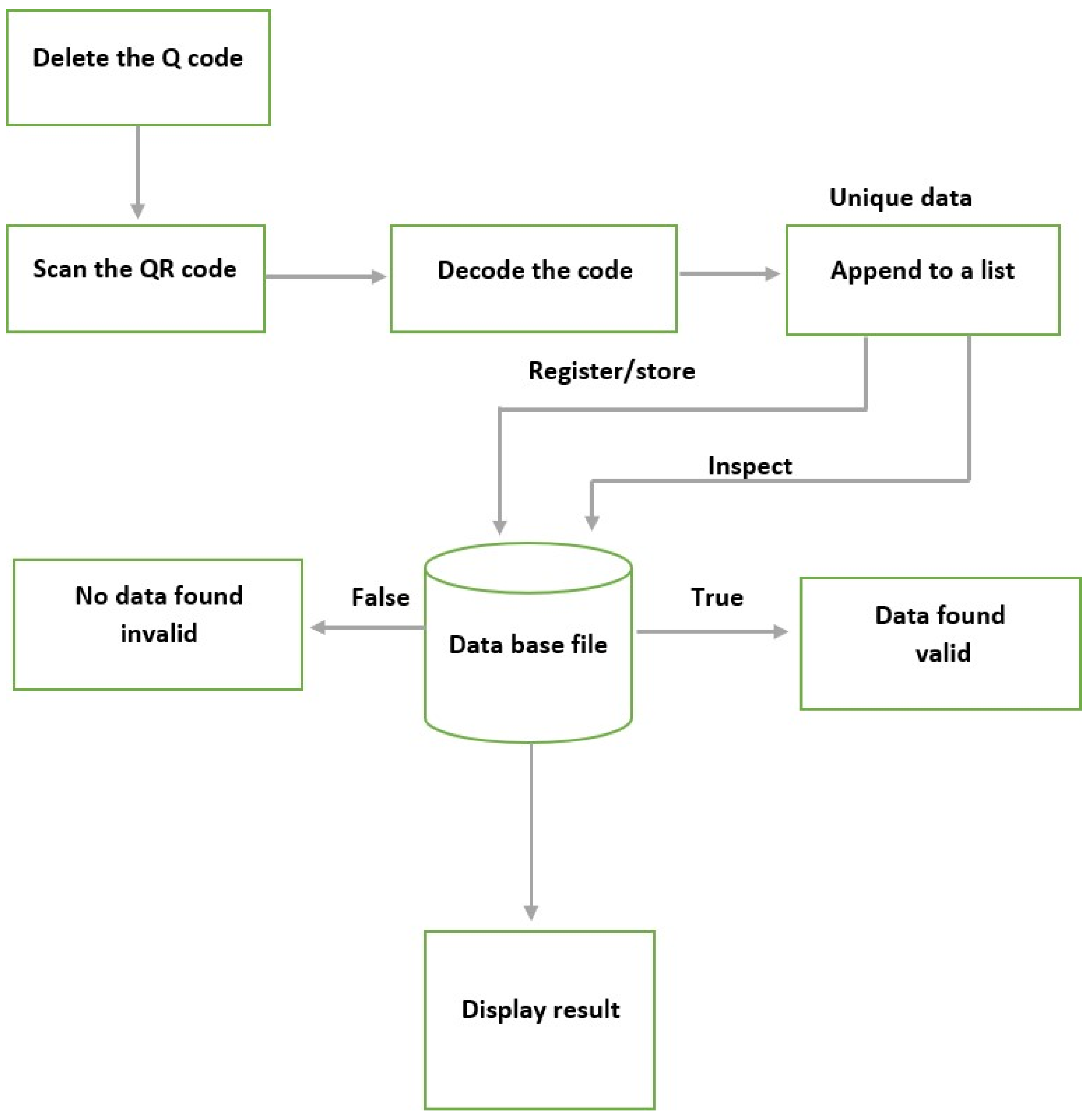


Fig 4.1.1: System architecture

4.2.1 Algorithm

Step 1: Open a empty python file and name the file.

Step 2: Import the required libraries.
1.OpenCV-Python.
2.Numpy.
3.Pyzbar.

Step 3: Webcam access using cv2 function - cv2.VideoCapture(0), here argument 0 because we are using internal camera of the system.

Step 4: Take input of the database file name.

Step 5: Create a list to store unique decoded data from scanning.

Step 6: Enter the choice according to what operation need to be performed.

Step 7: Open the database file in ‘r+’ mood.

Step 8: Store the each line of the file as a element into a list using readlines.split(\n) function.

Step 9: If choice is “I” – Inspection of the data.

Step 10: Enter number of scans to be performed.

Step 11: Read the frames through the webcam.

Step 12: Detecting the QR/Barcode in the frame and decoding.

Step 13: Displaying the bounding lines around the QR/Barcode.

Step 14: Displaying the decoded data on the frame.

Step 15: If decoded data is present in the database file then print valid else invalid.

Step 16: Display the real time video capturing frame.

Step 17: Close the database file.

Step 18: If choice is “R” – Registering the data.

Step 19: Step10 to Step13 .

Step 20: If the decoded data is not present in the list, append the decoded data into the list.

Step 21: Write the data which has been append to the list into the database file.

Step 22: Step 16 and Step 17 .

Step 23: If choice is not “I”, “R” or “E” – Invalid entry.

Step 24: Print Invalid choice and again ask to enter valid choice.

Step 25: If choice is “E” – Exit .

Step 26: Print the Exiting message.

Step 27: Terminate the program.

4.2.2 Flowchart of Proposed System

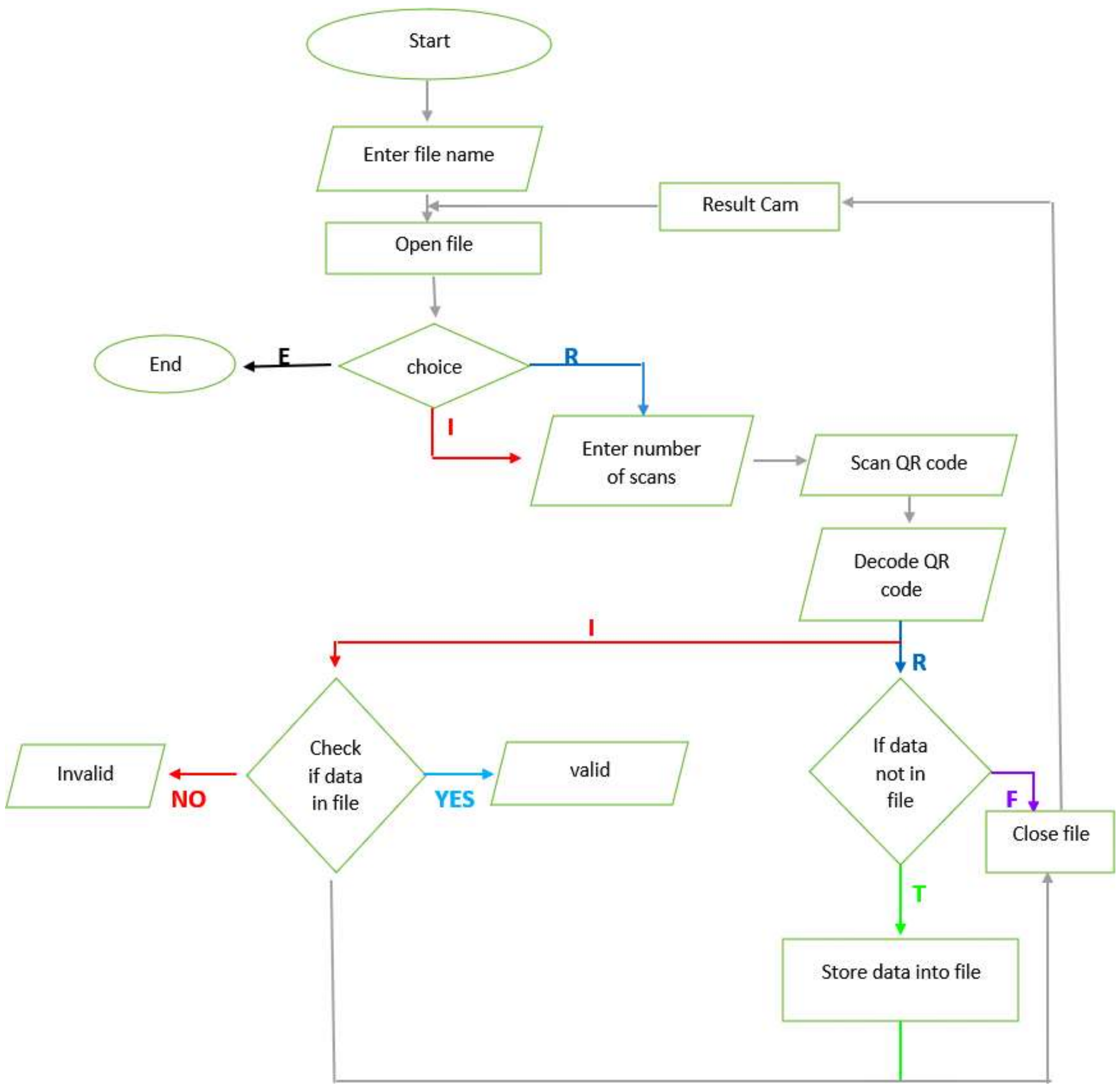


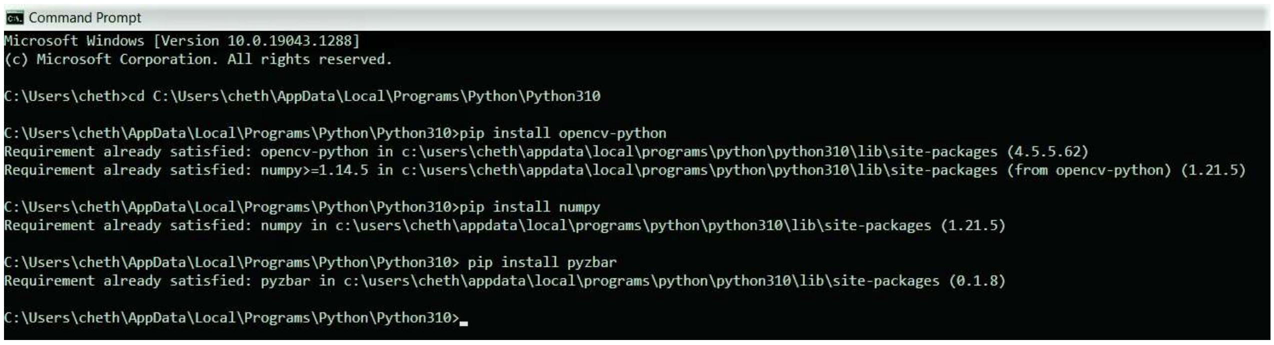
Fig 4.2.2: Flowchart of proposed system

CHAPTER 5

IMPLEMENTATION

5.1 System Modules

We are using Python 3.10 language to implement our system, So the system requires **‘OpenCV-Python’, ‘NumPy’ and ‘Pyzbar’** module packages to be installed to the path where python interpreter is located in the PC of the user.



```
Command Prompt
Microsoft Windows [Version 10.0.19043.1288]
(c) Microsoft Corporation. All rights reserved.

C:\Users\cheth>cd C:\Users\cheth\AppData\Local\Programs\Python\Python310

C:\Users\cheth\AppData\Local\Programs\Python\Python310>pip install opencv-python
Requirement already satisfied: opencv-python in c:\users\cheth\appdata\local\programs\python\python310\lib\site-packages (4.5.5.62)
Requirement already satisfied: numpy>=1.14.5 in c:\users\cheth\appdata\local\programs\python\python310\lib\site-packages (from opencv-python) (1.21.5)

C:\Users\cheth\AppData\Local\Programs\Python\Python310>pip install numpy
Requirement already satisfied: numpy in c:\users\cheth\appdata\local\programs\python\python310\lib\site-packages (1.21.5)

C:\Users\cheth\AppData\Local\Programs\Python\Python310> pip install pyzbar
Requirement already satisfied: pyzbar in c:\users\cheth\appdata\local\programs\python\python310\lib\site-packages (0.1.8)

C:\Users\cheth\AppData\Local\Programs\Python\Python310>_
```

Fig 5.1.1 Downloading packages

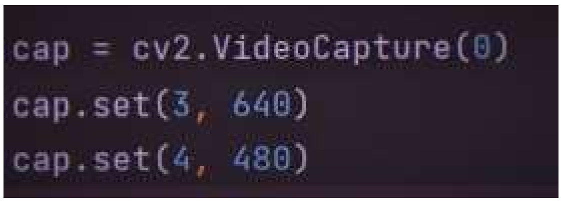
Cv2, numpy and pyzbar modules are imported to the program.

- **Cv2 -** It is module imported from the OpenCV-python package and it is an excellent tool for image processing and computer vision tasks. It is an open-source library that may be used for tasks such as face detection, objection tracking, landmark detection, and many others.

- **NumPy -** NumPy is a Python package that is used to work with arrays. It also includes functions for working with linear algebra, Fourier transforms, and matrices.

- **Pyzbar –** a pure Python library that reads 1-dimensional barcodes and QR codes using the zbar library, an open source software suite for reading bar codes from various sources, such as video streams, image files and raw intensity sensors.

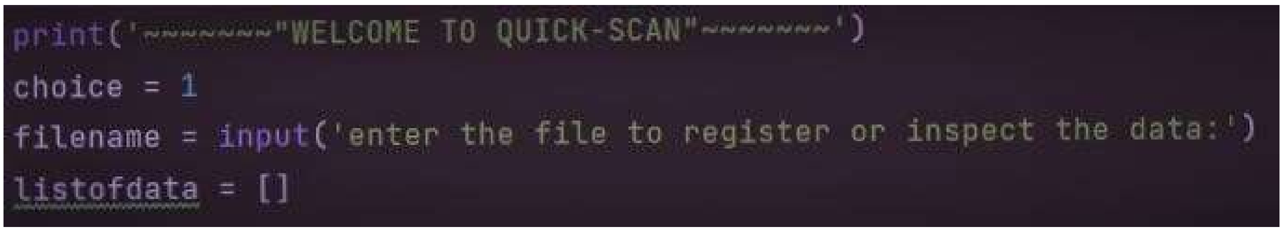
- **Implementing the code to capture the image and changing the video resolution**



```
cap = cv2.VideoCapture(0)
cap.set(3, 640)
cap.set(4, 480)
```

Here using cv2 function we setting the resolution of the camera and in the function call **cap.set()** parameters '3' and '4' indicates width and height of the frame and followed by corresponding length of width and height.

- **Implementing the code to take input of the database file**

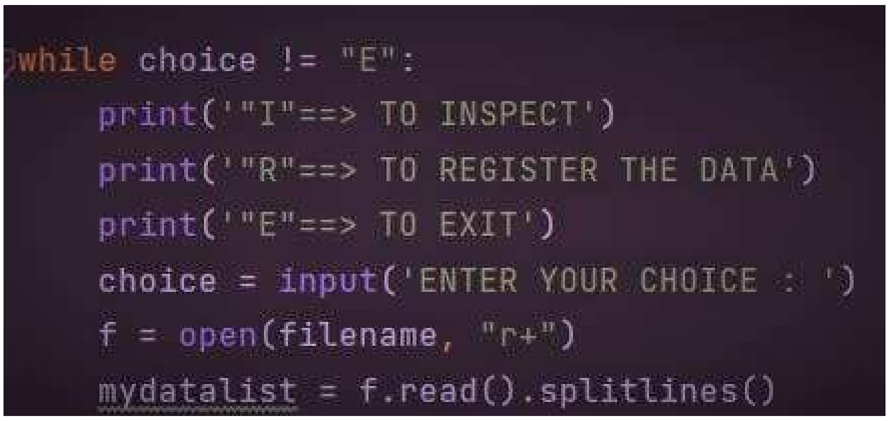


```
print('~~~~~~~~"WELCOME TO QUICK-SCAN"~~~~~~~~')
choice = 1
filename = input('enter the file to register or inspect the data:')
listofdata = []
```

listofdata is a variable object which holds the data of the scanned ticket before writing the data in the database file
choice is an object used to switch the operation to be performed in the program.

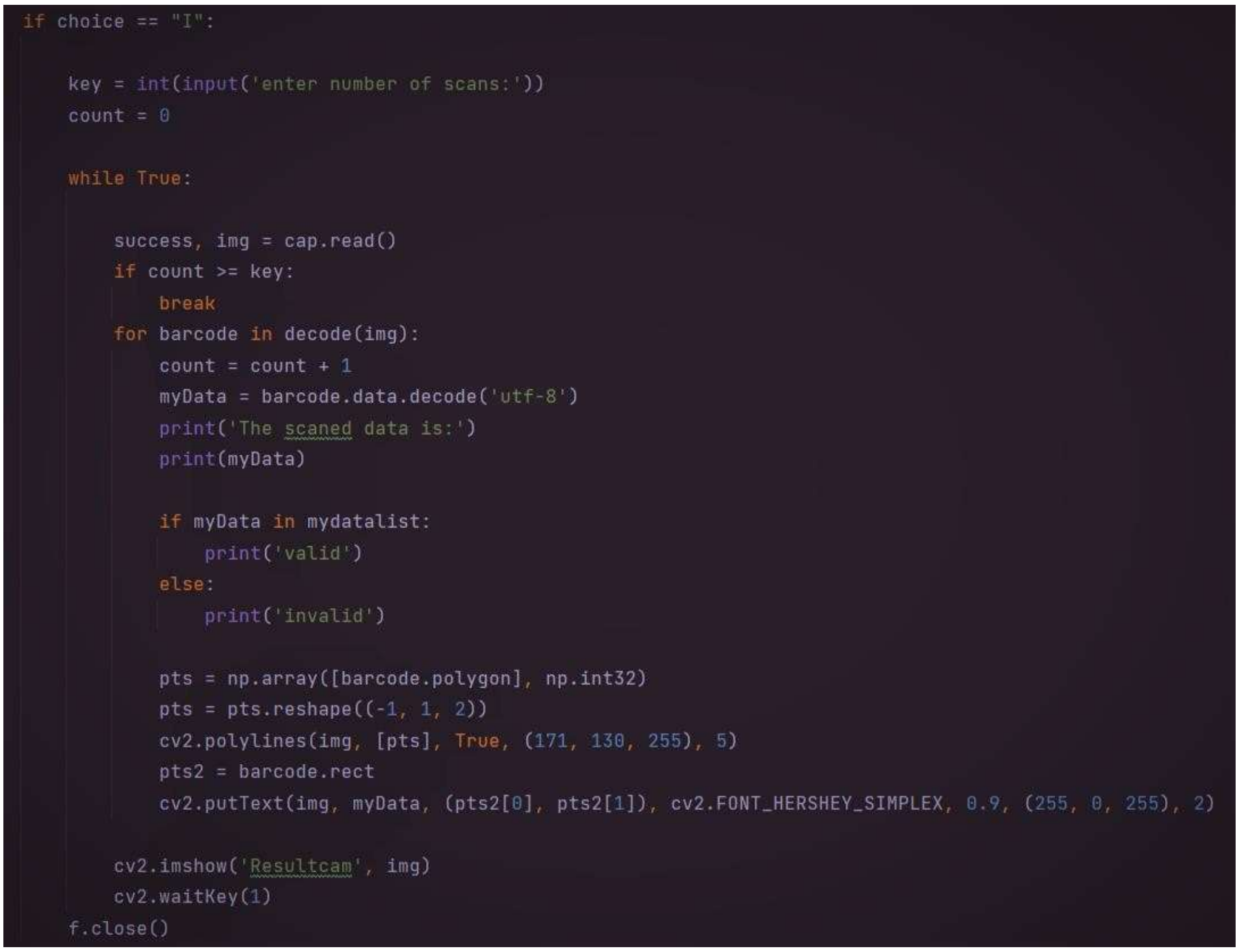
- **Implementing the code to make a choice of operation the user wants to make**

While loop is being used to make decision to choose which operation to be done
There are choices user can make 'I' to inspect the ticket's details by checking if the data is present in the database, 'R' to register the ticket's details data into the database file and 'E' to exit from the program
Here fille is opened in 'r+' mood where a file can be opened to read and write at a time.
If the file as any details in it then data in each line will be appended to a list object **mydatalist.**



```
while choice != "E":
    print('"I"==> TO INSPECT')
    print('"R"==> TO REGISTER THE DATA')
    print('"E"==> TO EXIT')
    choice = input('ENTER YOUR CHOICE : ')
    f = open(filename, "r+")
    mydatalist = f.read().splitlines()
```

- **Implementing the code if the choice is 'I' – Inspecting cycle**



```
if choice == "I":

    key = int(input('enter number of scans:'))
    count = 0

    while True:

        success, img = cap.read()
        if count >= key:
            break
        for barcode in decode(img):
            count = count + 1
            myData = barcode.data.decode('utf-8')
            print('The scaned data is:')
            print(myData)

            if myData in mydatalist:
                print('valid')
            else:
                print('invalid')

            pts = np.array([barcode.polygon], np.int32)
            pts = pts.reshape((-1, 1, 2))
            cv2.polylines(img, [pts], True, (171, 130, 255), 5)
            pts2 = barcode.rect
            cv2.putText(img, myData, (pts2[0], pts2[1]), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (255, 0, 255), 2)

        cv2.imshow('Resultcam', img)
        cv2.waitKey(1)
    f.close()
```

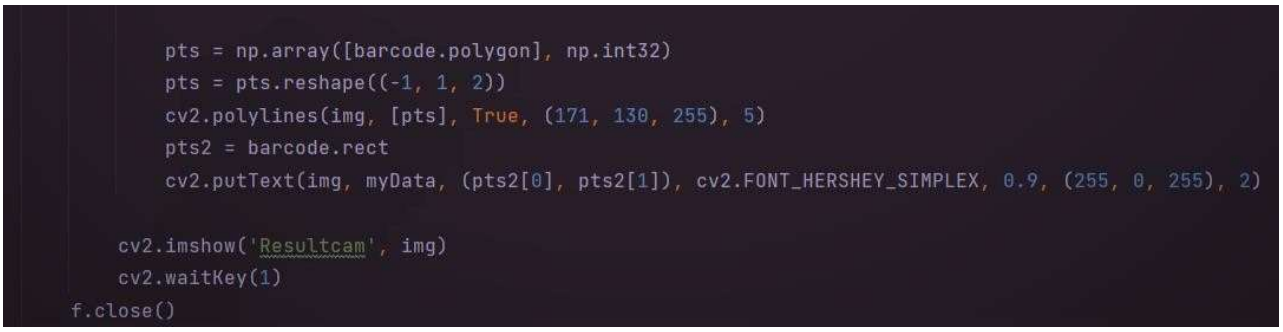
Here key is an object which takes input of number of scans which refers to number of times a QR code or Barcode to be detected and decoded.

Cap.read() is a function of OpenCV which returns a bool (True or False), it capture frame by frame and if the frame is read correctly it returns True else false. The function returns two values frame and Boolean to the objects success and img respectively.

Count is a object which holds the value ‘0’(int data type) initially if the value of the count is more and key the while loop will be terminated and control goes to choice selection. Count is incremented by 1 for every QR/barcode scanned and decoded in the program.

barcode.data.decode('utf-8') is a function call to decode the scanned QR/Barcode into binary followed by character and digits, it returns the data to the object myData
‘utf-8’ is a variable length method of encoding Unicode characters such as Arabic, Russian, Japanese, Hindi, or Thai characters. myData id printed to show the details of the tickets data.

To inspect the details of the tickets whether it is present in database are not membership operator ‘**in**’ is used and the result ‘valid’ or ‘invalid’ is printed correspondingly.



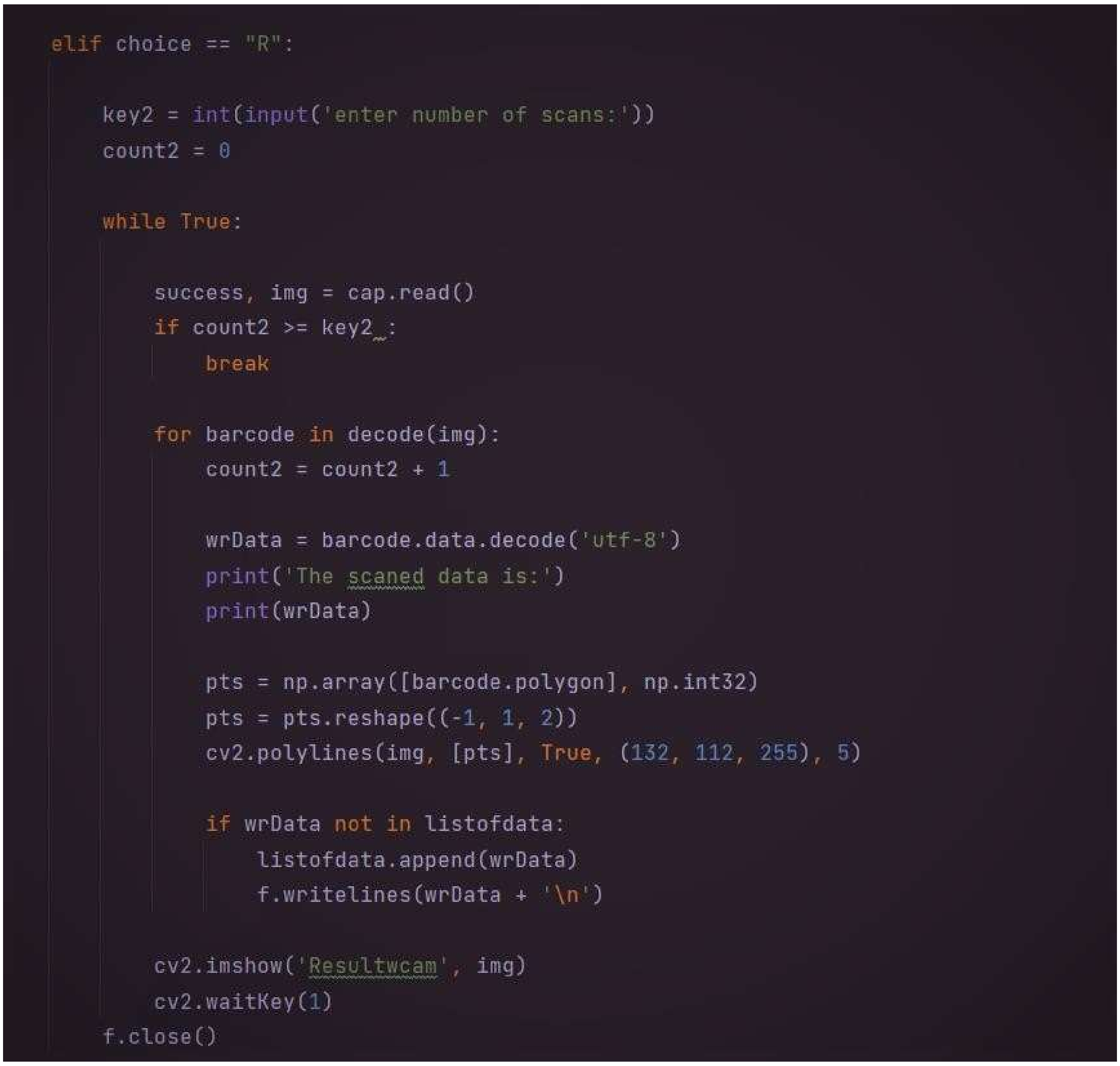
```
            pts = np.array([barcode.polygon], np.int32)
            pts = pts.reshape((-1, 1, 2))
            cv2.polylines(img, [pts], True, (171, 130, 255), 5)
            pts2 = barcode.rect
            cv2.putText(img, myData, (pts2[0], pts2[1]), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (255, 0, 255), 2)

        cv2.imshow('Resultcam', img)
        cv2.waitKey(1)
    f.close()
```

In the above code is used to construct a boundary on the edges of the detected QR/barcode using numPy module and display the bounded lines on the visual output using **cv2.polylines()** function and the decoded data is also displayed in the visual output by using **cv2.putText()** function.

cv2.imshow() is a function used to display the visual output as first argument name as the object. cv2.waitkey(1) is a function used as a time delay between the next scan cycle where ‘1’ is time 1 millisecond.

Database file is closed after inspection is done by using fileobject ‘f’.

➢ **Implementing the code if the choice is ‘R’ – Registering cycle**



```
elif choice == "R":

    key2 = int(input('enter number of scans:'))
    count2 = 0

    while True:

        success, img = cap.read()
        if count2 >= key2 :
            break

        for barcode in decode(img):
            count2 = count2 + 1

            wrData = barcode.data.decode('utf-8')
            print('The scaned data is:')
            print(wrData)

            pts = np.array([barcode.polygon], np.int32)
            pts = pts.reshape((-1, 1, 2))
            cv2.polylines(img, [pts], True, (132, 112, 255), 5)

            if wrData not in listofdata:
                listofdata.append(wrData)
                f.writelines(wrData + '\n')

        cv2.imshow('Resultwcam', img)
        cv2.waitKey(1)
    f.close()
```

Here key2 and count2 does the same task as the key and count in the inspecting cycle.

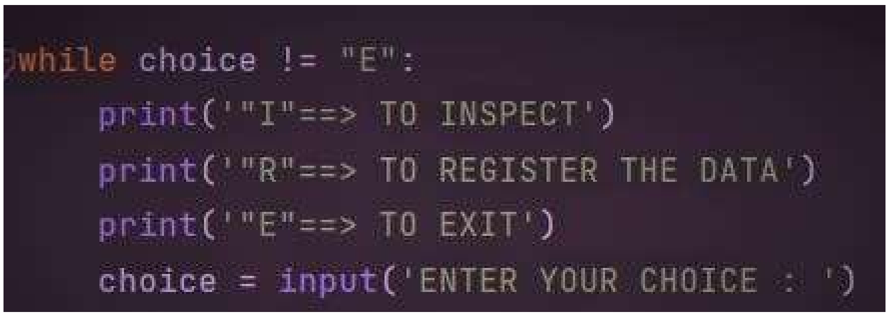
wrData is the object which holds the decoded data of the tickets scanned and decoded data is printed.

Using **reshape()** function of numPy and cv2.polylines() function of OpenCV boundary lines are displayed on the visual output.

Membership operator 'not in' is used to check if the scanned and decoded data is present in a list '**listofdata**', only unique data will be append to list and the unique data will be written into the database file.

The visual output of the capturing of the webcam will be displayed by the function **cv2.imshow()** under the name 'Resultwcam'. The file will be closed if the Register cycle is completed.

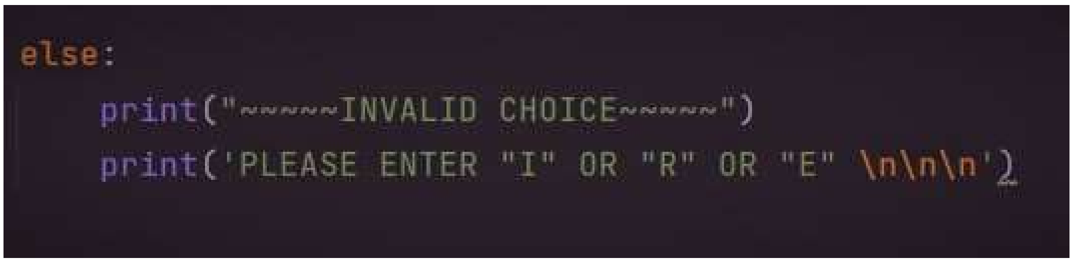
- **Implementing the code if the choice is 'E' – End the program**



```
while choice != "E":
    print('"I"==> TO INSPECT')
    print('"R"==> TO REGISTER THE DATA')
    print('"E"==> TO EXIT')
    choice = input('ENTER YOUR CHOICE : ')
```

when choice =='E' the while loop condition is not satisfied and the loop breaks to terminate the entire program.

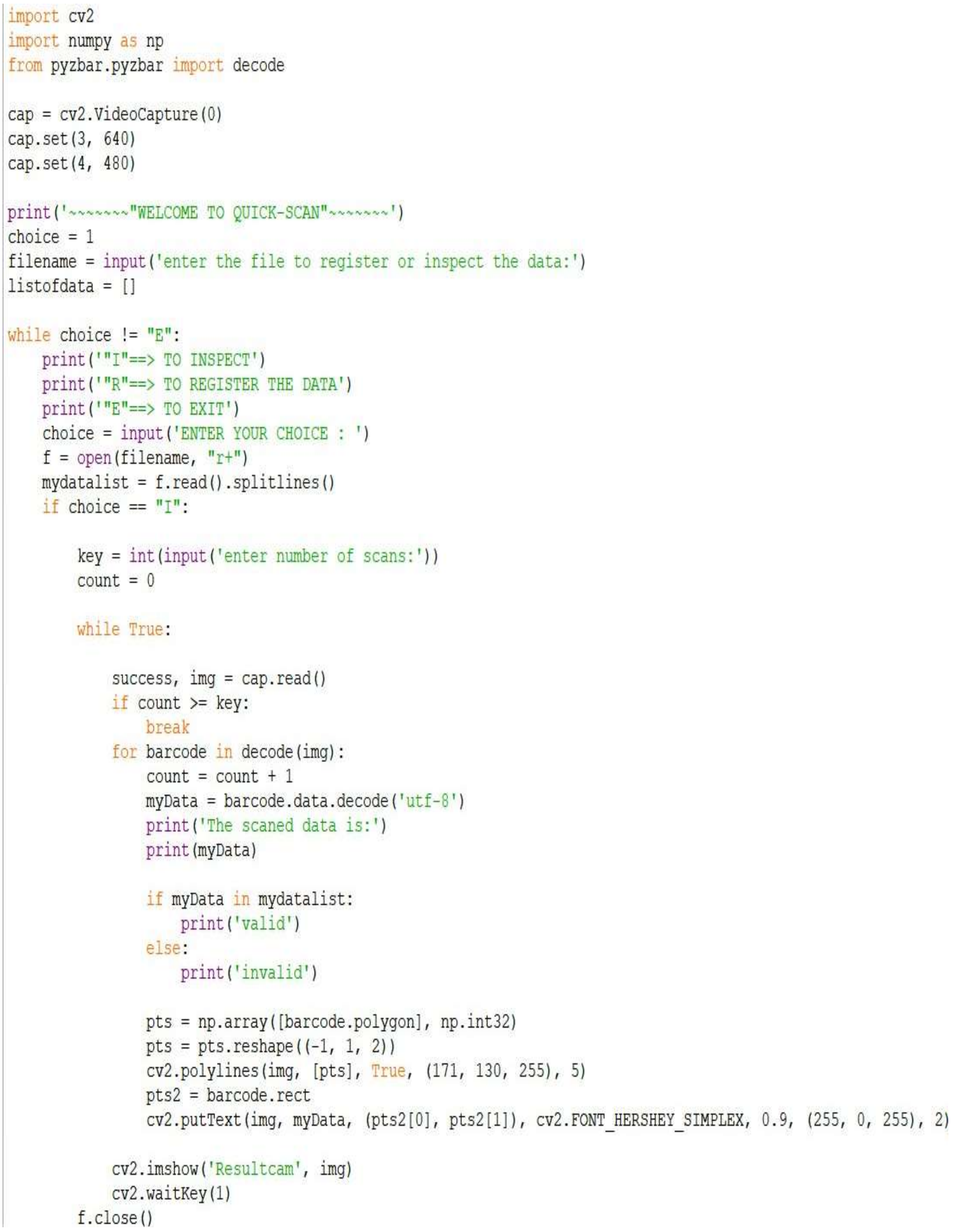
- **Implementing the code if the choice is other than 'I', 'R' and 'E'**



```
else:
    print("~~~~~INVALID CHOICE~~~~~")
    print('PLEASE ENTER "I" OR "R" OR "E" \n\n\n')
```

The control goes to **'else'** block to print the 'INVALID CHOICE' string and again due the while loop again the controls goes to 'CHOICE' making block to take the input from the user. The same codes will be executed until the choice input is 'E'.

❖ FINAL PROGRAM CODE



```python
import cv2
import numpy as np
from pyzbar.pyzbar import decode

cap = cv2.VideoCapture(0)
cap.set(3, 640)
cap.set(4, 480)

print('~~~~~~~"WELCOME TO QUICK-SCAN"~~~~~~~')
choice = 1
filename = input('enter the file to register or inspect the data:')
listofdata = []

while choice != "E":
    print('"I"==> TO INSPECT')
    print('"R"==> TO REGISTER THE DATA')
    print('"E"==> TO EXIT')
    choice = input('ENTER YOUR CHOICE : ')
    f = open(filename, "r+")
    mydatalist = f.read().splitlines()
    if choice == "I":

        key = int(input('enter number of scans:'))
        count = 0

        while True:

            success, img = cap.read()
            if count >= key:
                break
            for barcode in decode(img):
                count = count + 1
                myData = barcode.data.decode('utf-8')
                print('The scaned data is:')
                print(myData)

                if myData in mydatalist:
                    print('valid')
                else:
                    print('invalid')

                pts = np.array([barcode.polygon], np.int32)
                pts = pts.reshape((-1, 1, 2))
                cv2.polylines(img, [pts], True, (171, 130, 255), 5)
                pts2 = barcode.rect
                cv2.putText(img, myData, (pts2[0], pts2[1]), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (255, 0, 255), 2)

            cv2.imshow('Resultcam', img)
            cv2.waitKey(1)
        f.close()
```

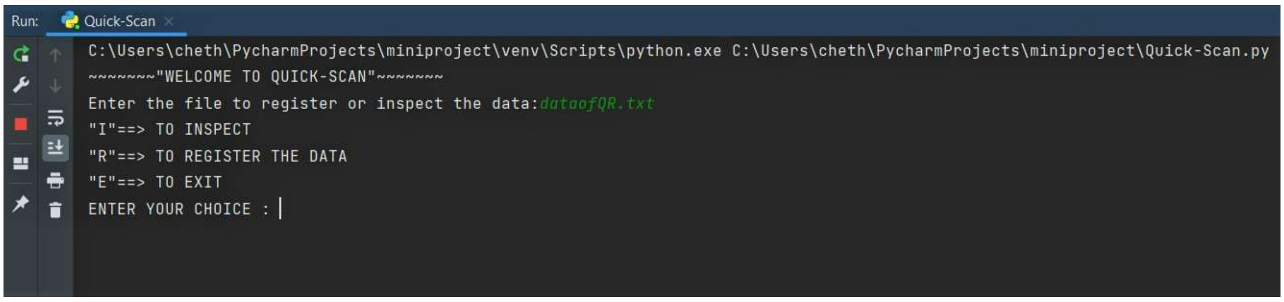
```
elif choice == "R":

    key2 = int(input('enter number of scans:'))
    count2 = 0

    while True:

        success, img = cap.read()
        if count2 >= key2 :
            break

        for barcode in decode(img):
            count2 = count2 + 1

            wrData = barcode.data.decode('utf-8')
            print('The scaned data is:')
            print(wrData)

            pts = np.array([barcode.polygon], np.int32)
            pts = pts.reshape((-1, 1, 2))
            cv2.polylines(img, [pts], True, (132, 112, 255), 5)

            if wrData not in listofdata:
                listofdata.append(wrData)
                f.writelines(wrData + '\n')

        cv2.imshow('Resultwcam', img)
        cv2.waitKey(1)
    f.close()

elif choice == "E":
    print("Exiting from the program!!")

else:
    print("~~~~~INVALID CHOICE~~~~~")
    print('PLEASE ENTER "I" OR "R" OR "E" \n\n\n')
```

CHAPTER 6

EXPERIMENTAL RESULTS

6.1 Outcome of Proposed System

When the Quick-Scan program is run on Pycharm IDE Python compiler under the Run terminal the Outcome of the program will be displayed.

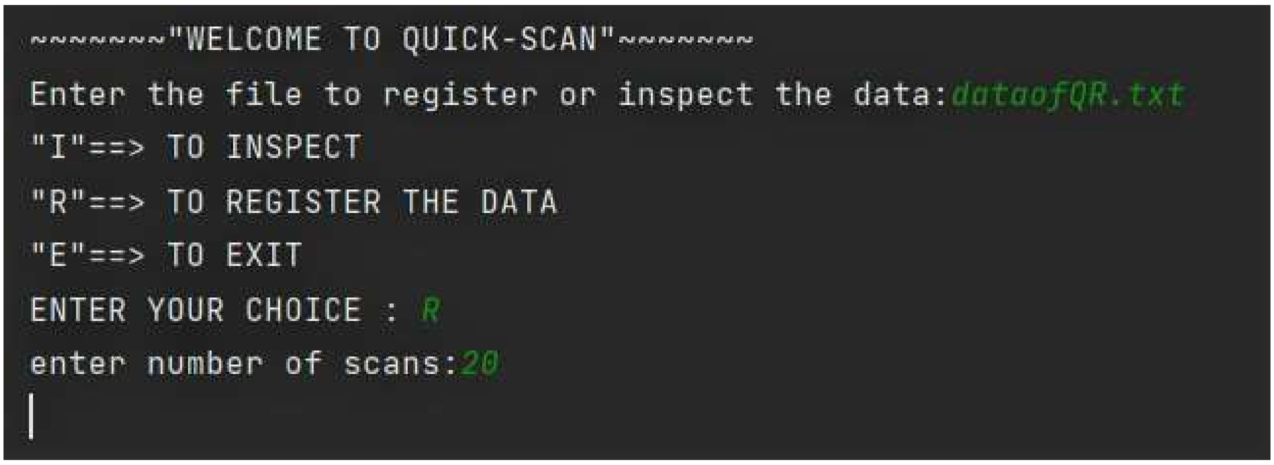


```
C:\Users\cheth\PycharmProjects\miniproject\venv\Scripts\python.exe C:\Users\cheth\PycharmProjects\miniproject\Quick-Scan.py
~~~~~~~"WELCOME TO QUICK-SCAN"~~~~~~~
Enter the file to register or inspect the data:dataofQR.txt
"I"==> TO INSPECT
"R"==> TO REGISTER THE DATA
"E"==> TO EXIT
ENTER YOUR CHOICE : |
```

User has to enter the name of the database file and the file must be present in the same path as the program file.

User has to choose the operation to be done in the program, there are 3 choice a user can make:
“I”➔Inspect the tickets.
“R”➔Register the tickets into the database file.
“E”➔Exit from the program.

If the choice is “R”:



```
~~~~~~~"WELCOME TO QUICK-SCAN"~~~~~~~
Enter the file to register or inspect the data:dataofQR.txt
"I"==> TO INSPECT
"R"==> TO REGISTER THE DATA
"E"==> TO EXIT
ENTER YOUR CHOICE : R
enter number of scans:20
|
```

Enter number of scans to be done, the time delay is 1milisecond so same QR or barcode will be scanned more than once.
The frame window of the webcam will be opened under the name **Resultwcam**

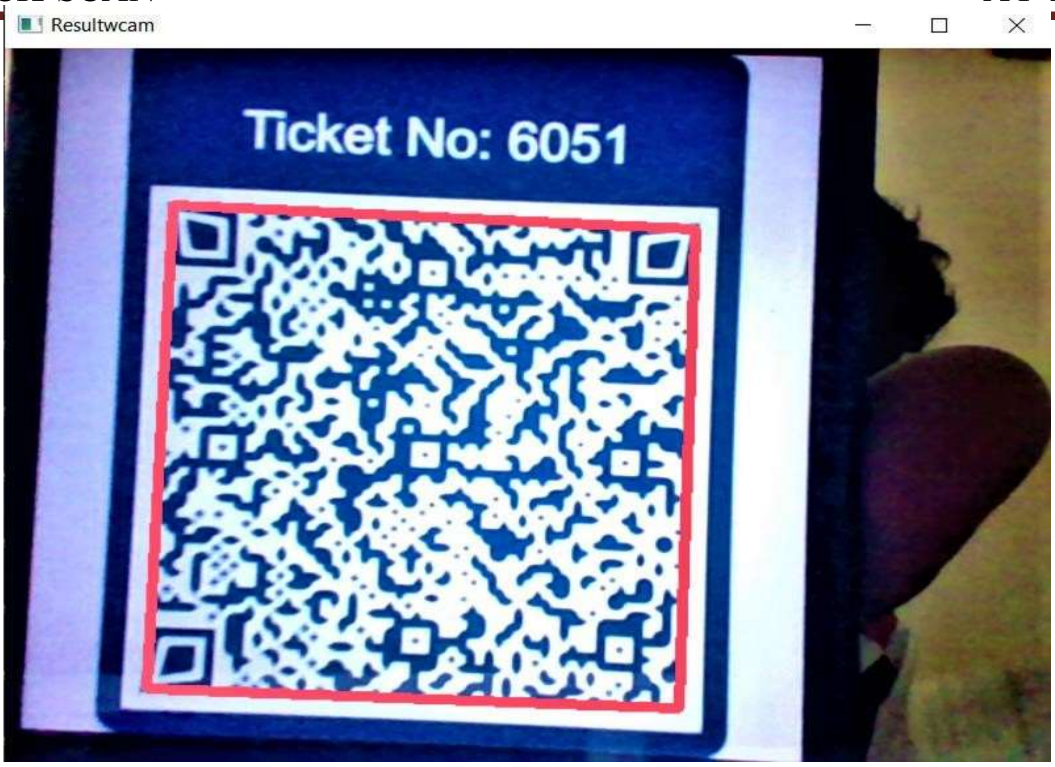
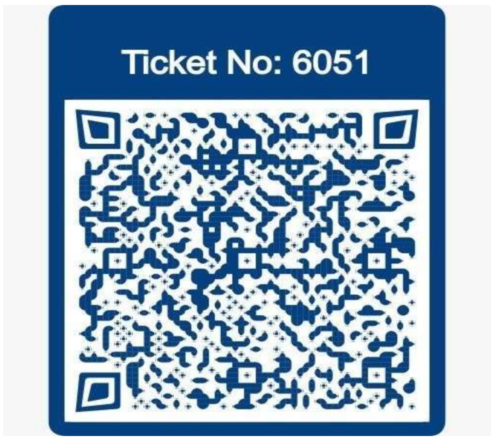


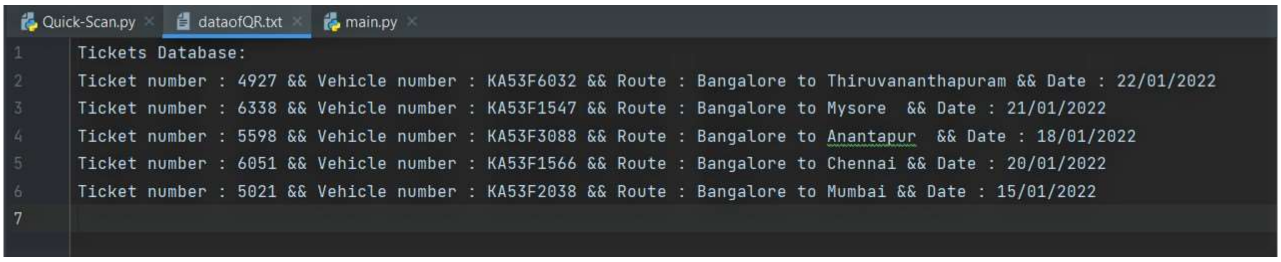
Fig 6.1.1 Registering the tickets

Scan the QR code present in the Tickets to register in database file.



Any number of tickets can be scanned to register in the database.

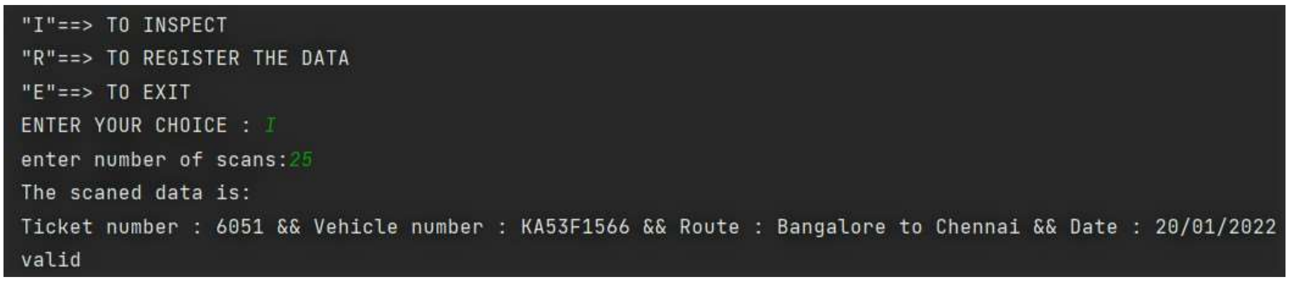
After scanning the tickets the data will be stored in the file named **dataofQR.txt.**



```
Tickets Database:
Ticket number : 4927 && Vehicle number : KA53F6032 && Route : Bangalore to Thiruvananthapuram && Date : 22/01/2022
Ticket number : 6338 && Vehicle number : KA53F1547 && Route : Bangalore to Mysore  && Date : 21/01/2022
Ticket number : 5598 && Vehicle number : KA53F3088 && Route : Bangalore to Anantapur  && Date : 18/01/2022
Ticket number : 6051 && Vehicle number : KA53F1566 && Route : Bangalore to Chennai && Date : 20/01/2022
Ticket number : 5021 && Vehicle number : KA53F2038 && Route : Bangalore to Mumbai && Date : 15/01/2022
```

After the ‘Registering cycle’ is completed again the user can make the choice to register the tickets or we can inspect the tickets.

If choice is “I”:



```
"I"==> TO INSPECT
"R"==> TO REGISTER THE DATA
"E"==> TO EXIT
ENTER YOUR CHOICE : I
enter number of scans:25
The scaned data is:
Ticket number : 6051 && Vehicle number : KA53F1566 && Route : Bangalore to Chennai && Date : 20/01/2022
valid
```

If the scanned ticket details present in the database file then the above output will be printed, i.e. Valid ticket.

Resultcam window will be opened to show webcam frame.

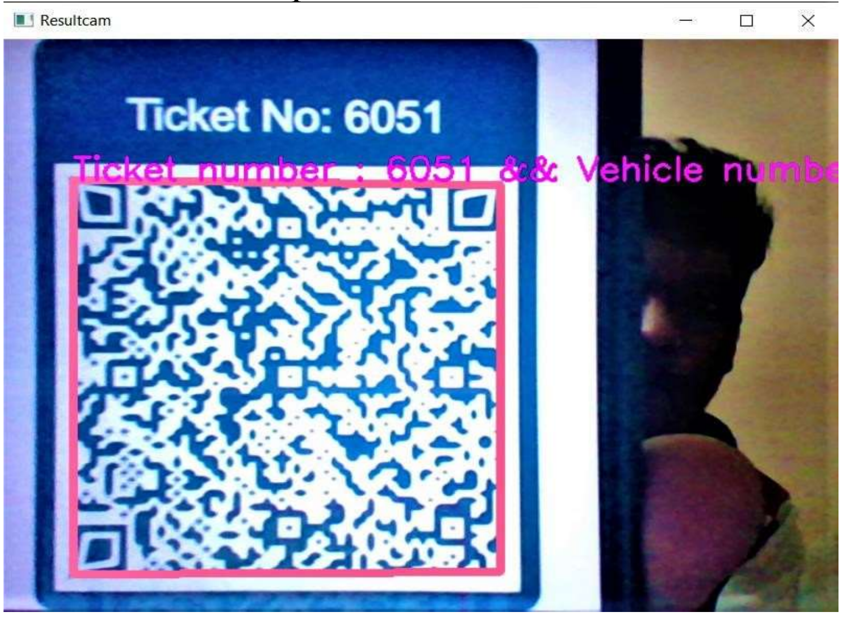
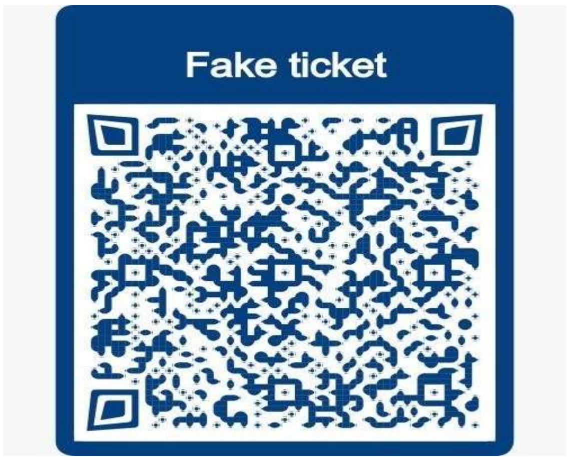


Fig 6.1.2 Inspecting the ticket

When the scanned ticket is counterfeit or not registered in the database file.



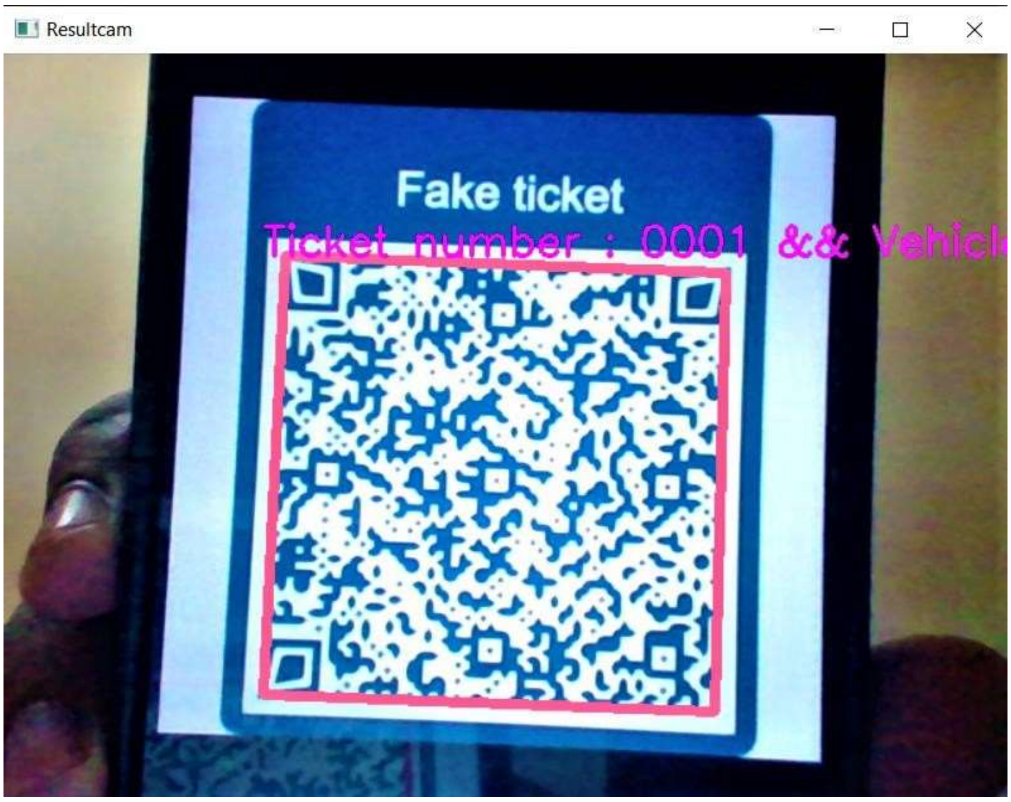


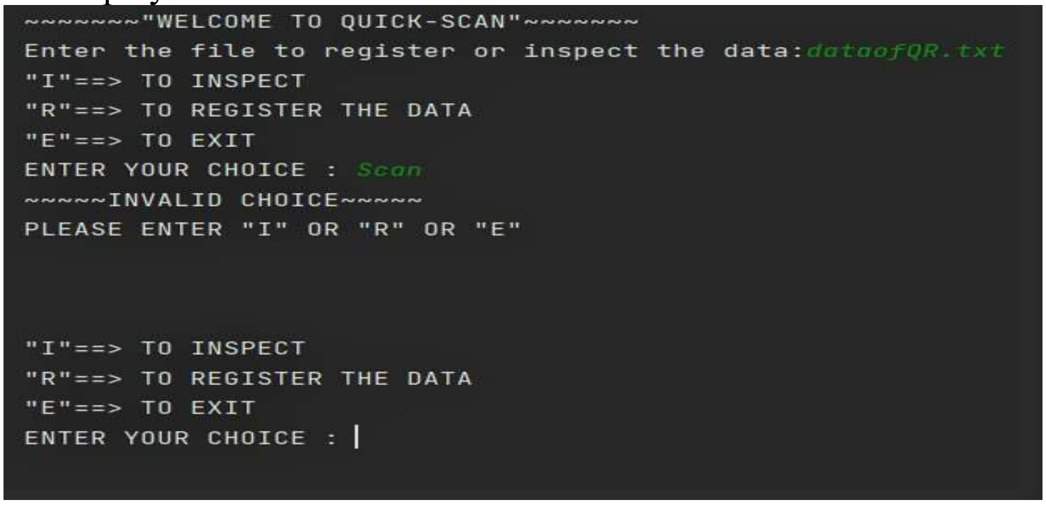
Fig 6.1.3 Inspecting Fake ticket

If the ticket scanned is fake/counterfeit or not valid for the day then the program shows invalid message along with the decoded message.

```
The scaned data is:
Ticket number : 0001 && Vehicle number : KA53F5732 && Route : Bangalore to Kolar && Date : 23/01/2022
invalid
```

If the choice entered is not valid:

Other than ‘I’,’R’ and ‘E’ choice any other input entered is invalid choice and below output will be displayed.

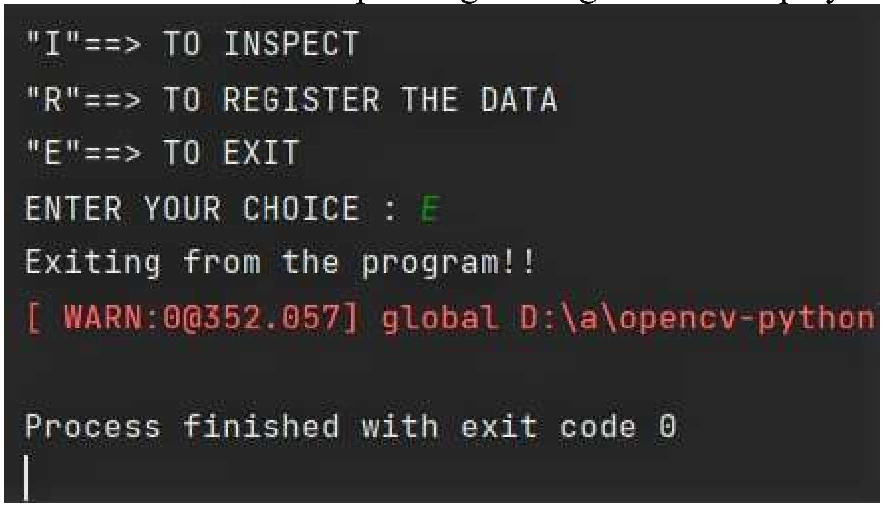


```
~~~~~~~"WELCOME TO QUICK-SCAN"~~~~~~~
Enter the file to register or inspect the data:dataofQR.txt
"I"==> TO INSPECT
"R"==> TO REGISTER THE DATA
"E"==> TO EXIT
ENTER YOUR CHOICE : Scan
~~~~~INVALID CHOICE~~~~~
PLEASE ENTER "I" OR "R" OR "E"



"I"==> TO INSPECT
"R"==> TO REGISTER THE DATA
"E"==> TO EXIT
ENTER YOUR CHOICE : |
```

This output will be displayed until valid choice is entered by the user.

If the choice is “E”:

The will be terminated and the corresponding message will be displayed as shown below.



```
"I"==> TO INSPECT
"R"==> TO REGISTER THE DATA
"E"==> TO EXIT
ENTER YOUR CHOICE : E
Exiting from the program!!
[ WARN:0@352.057] global D:\a\opencv-python

Process finished with exit code 0
|
```

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 Conclusion

This project is about incorporating the scanning technology to get the digital data and registering the fetched data into the database and inspecting the data of scanned tickets if its data present in database file of the transit companies.

Registration of the ticket's details data into a database file by using QR/Barcode scanner is more productive than the manual registration of the data into a file, similarly inspecting of the tickets in the mass transit is done using the scanner if the tickets data is present in the database file then it is a valid ticket else it would be a counterfeit ticket.

Ticket inspectors work is made more effective and productive in mass public transit by using 'Quick-Scan' system, rather than hassle job inspecting tickets manually.

7.2 Future Enhancement

Our system has compatibility for future implementation, with small changes in the program our system can be used for taking attendance of the employees by scanning their ID cards and many other applications can be enhanced.

A mobile hardware tool can be developed to incorporate our software and it can be carried in the transit to inspect the passengers travelling.

An smartphone app can be implemented for passengers use where they can get their ticket information digitally.

REFERENCES

[1] OpenCV with Python: A basic approach by Dr. Panchanand Jha (Author), Dr. Bibhuti Bhusan Biswal (Contributor)

[2] NumPy for Beginners: first Step to learn Data Science by Preeti Saraswat

[3] https://towardsdatascience.com/build-your-own-barcode-and-qrcode-scanner-using-python-8b46971e719e

[4] https://www.youtube.com/watch?v=SrZuwM705yE&t=655s

[5] https://www.geeksforgeeks.org/reading-generating-qr-codes-python-using-qrtools/

[6] https://en.wikipedia.org