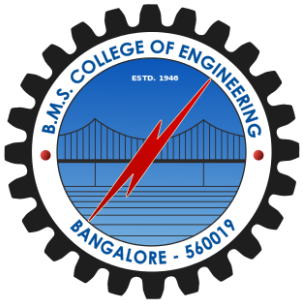


BMS COLLEGE OF ENGINEERING

(Autonomous College under VTU)
Bull Temple Road, Basavanagudi, Bangalore - 560019

Department of Information Science and Engineering

Submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering in Information Science & Engineering of Visvesvaraya Technological University, Belgaum

for the course

Computer Networks and Security [16IS6DCCNS]

By

PARAS NARENDRANATH [1BM16IS063]
PRAKHAR CHAUBE [1BM16IS066]

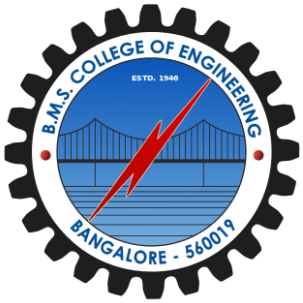
on the topic

Remote Automated RC Scanning module

Faculty In-Charge

Anitha H. M.
Faculty, Dept. of ISE

Jan – Jun 2019



BMS COLLEGE OF ENGINEERING

(Autonomous College under VTU)
Bull Temple Road, Basavanagudi, Bangalore - 560019

Department of Information Science and Engineering

CERTIFICATE

Certified that the Project has been successfully presented at B.M.S College Of Engineering by **PARAS NARENDRANATH** bearing USN: 1BM16IS063 and **PRAKHAR CHAUBE** bearing USN: 1BM16IS066 in partial fulfillment of the requirements for the VI Semester degree in Bachelor of Engineering in Information Science & Engineering of Visvesvaraya Technological University, Belgaum as a part of Self Study in Computer Networks and Security(16IS6DCCNS) during academic year 2018-2019.

Faculty Name – Anitha H. M.
Designation – Faculty
Department of ISE, BMSCE

Problem Statement

To develop a completely Remote and Automized Scanning module combined with a Storage system, that implements OCR to extract text from a scanned RC Card and stores the structured data into a Database.

Introduction

In view of the world switching to automization of all manual processes, we've come up with a solution that can help Vehicle insurance companies automize the process of collection vehicular details of its customers.

The RC card plays a vital role in identification of vehicles and their subsequent information obtained from the same can better help the company come up with policies with respect to these vehicles.

We have developed an automized scanning module that functions as an app on the customer end. The app plays the role of relaying video feed to the server, which processes this to identify the RC card present in these frames. It's subsequent retrieval is done and data from the obtained image is processed using Tesseract OCR to obtain the data in digital text form. Well structured Regular Expressions are created to correctly classify the data in the image.

Objectives

1. To develop an **Android App** for the customer that relays a video feed through network at a particular port.

2. Setting up a **Database environment** to store data of the customer RC cards

3. To setup a **Web Application** for the company to monitor the data present in the database.

4. Setup of a **Django Server** to remotely process all the data being sent to it from the customer app.

Technology Requirements

- **OpenCV**

  OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV is used to detect the RC card from the background and extract only that image.

- **Django**

  Is a Python Web Framework that allows building of the Web application and runs the processing code on the server.

- **Android Studio**

  Android App Development is implemented using Android Studio, to create an App that relays video feed from the camera to a dedicated socket on the network.

- **SQLite**

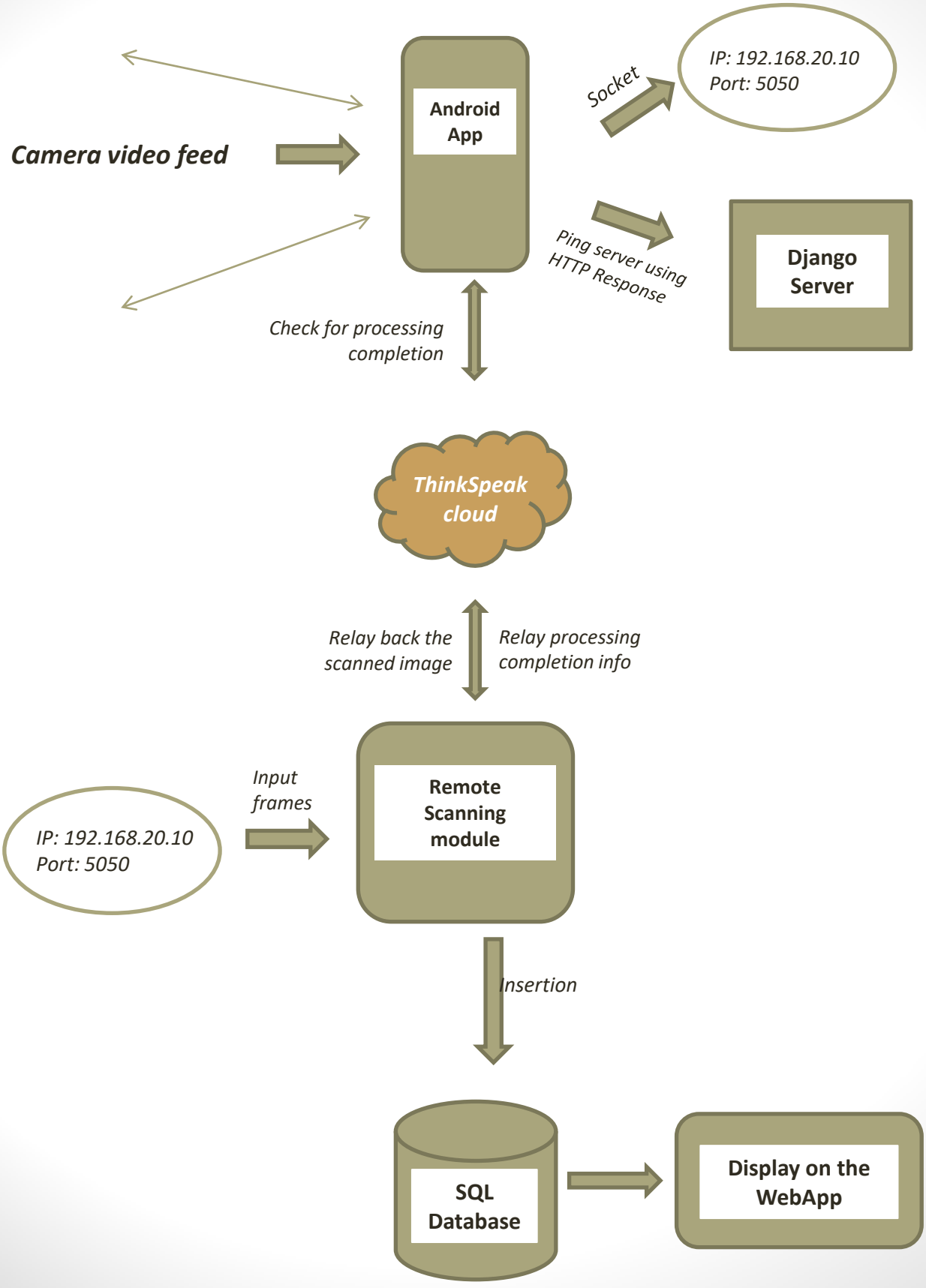
  Data is stored in a SQLite Database, connected to the servers.

- **Tesseract by Google**

  The pytesseract api was used to perform Optical Character Recognition.

- Programming Languages used:
  1. Python
  2. Java

Data Flow


Camera video feed
Android App
Socket
IP: 192.168.20.10
Port: 5050
Ping server using HTTP Response
Django Server
Check for processing completion
ThinkSpeak cloud
Relay back the scanned image
Relay processing completion info
IP: 192.168.20.10
Port: 5050
Input frames
Remote Scanning module
Insertion
SQL Database
Display on the WebApp

Implementation

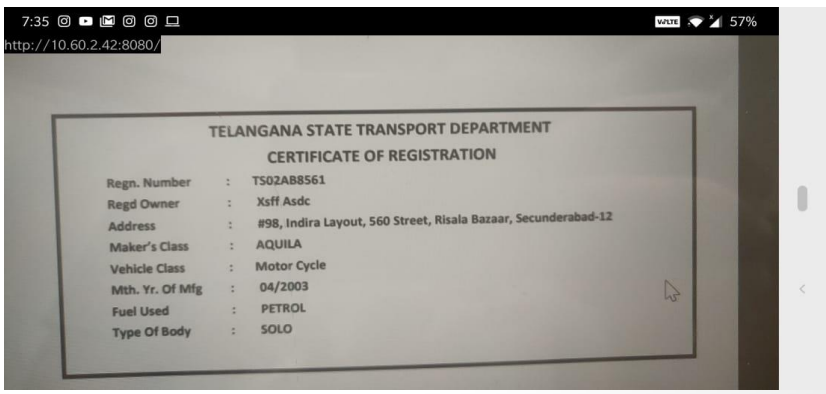
❖ Android App (RC Scanner)

- The app opens up the camera on the device and begins recording. The frames are retrieved, buffered and compressed into JPEG style frames.
- A server socket is opened at the device's IP and port:8080. These JPEG frames are then sent across this socket and are hosted at the corresponding URL. In this case, ***http://10.60.2.42:8080/***

```
private void acceptAndStream() throws IOException {
    ServerSocket serverSocket = null;
    Socket socket = null;
    DataOutputStream stream = null;
    try {
        serverSocket = new ServerSocket( port: 8080);
        serverSocket.setSoTimeout(1000 /* milliseconds */);
        do {
            try {
                socket = serverSocket.accept();
            }
            catch (final SocketTimeoutException e) {
                if (!mRunning) {
                    return;
                }
            }
        } while (socket == null);

        serverSocket.close();
        serverSocket = null;
        stream = new DataOutputStream(socket.getOutputStream());
        stream.writeBytes(HTTP_HEADER);
        stream.flush();
```

Hosting URL ⇨



❖ Django server

The server has *4 functionalities* :

1. **Process images to pick the best:**

- A single channel of an image (grayscale) and convolve it with a *3 x 3* kernel. The variance of the kernel is found

- If an image contains ***high*** *variance* then there is a wide spread of responses, and therefore, an in-focus image. But if there is ***very low*** *variance*, then there is a tiny spread of responses, indicating there are very little edges in the image. As we know, ***the more an image is blurred, the less edges there are*****.**

- The variance of all the images are found and the image with the ***max value of variance*** (ie., least blurry) is picked.

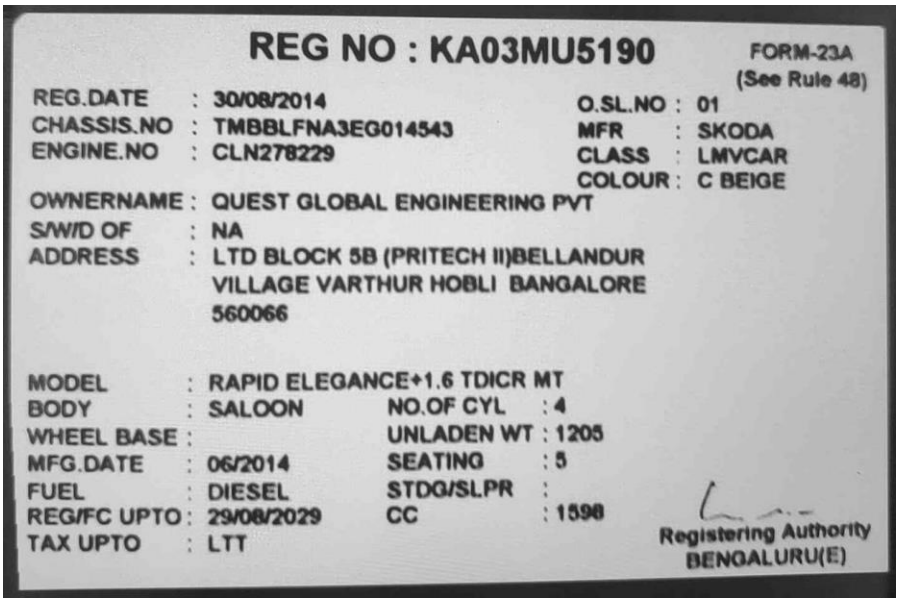


Fig. 1 After processing

2. **Identification of the RC card from its background:**

- **Thresholding**: Simple Thresholding is implemented on a grayscale image over a particular threshold value.
- **Dilation**: The boundaries are enhanced using dilation. It is useful in joining broken parts of an object
- **Contouring**: The required image is obtained based on its boundary points based on the contour constraints
- **Overlay**: The image obtained is superimposed on the original image to obtain the precise boundaries

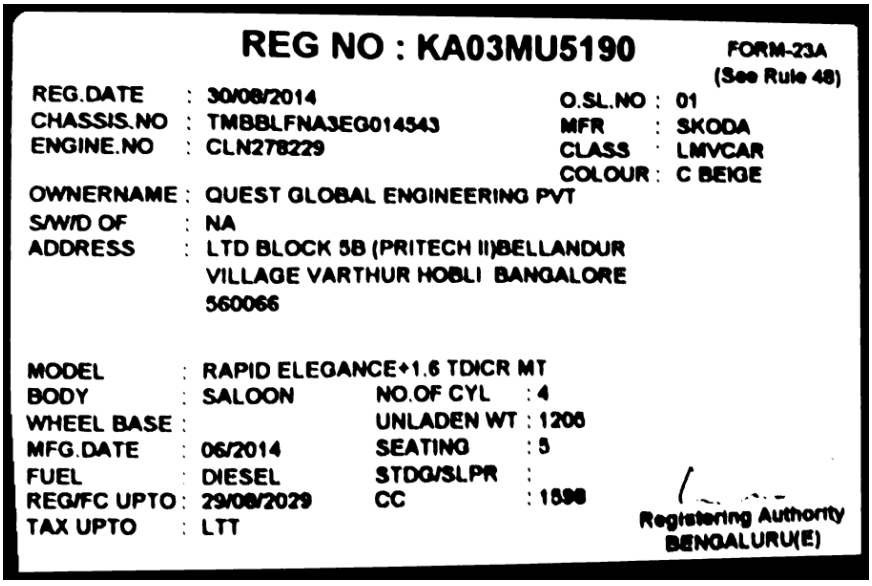


Fig 2. After Thresholding

3. **Extract the text features using TESSERACT:**

- **Tesseract** is an optical character recognition (OCR) system that converts image documents into text documents. It is a open-source software that is considered one of the most accurate OCR engines currently available

- *Pytesseract api* is fed the final processed image and the extracted text from the image is returned.

- Error correction is done on the strings to ensure presence of consistent data.

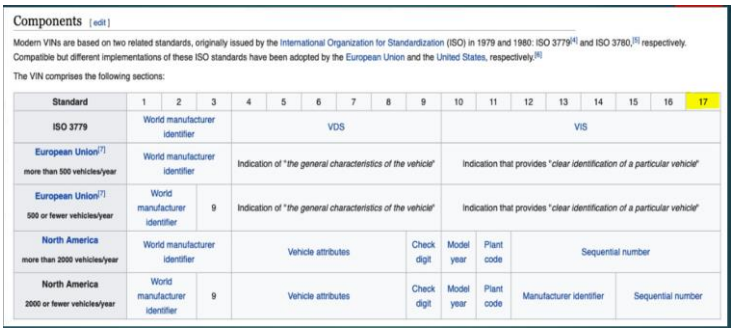
```
In [6]: pytesseract.image_to_string(Image.open('rv.jpeg'),
lang='eng')
Out[6]: 'REG.DATE\nCHASSIS.NO :\nENGINE.NO\n\nOWNERNAME :\nSAID OF
\nADDRESS\n\nMODEL\nBODY\n\nWHEEL BASE :\nMFG.DATE\nFUEL :\nREG/FC
UPTO:\nTAX UPTO\n\nREG NO : KA03MU5190 FORM-23A\n\n(See Rule 48)\n>
30/08/2014 O.SL.NO : 01\nTMBBLFNASEGO14543 MFR: SKODA\n: CLN278229
CLASS - LMVCAR\n\nCOLOUR: C BEIGE\nQUEST GLOBAL ENGINEERING PVT\n\n:
NA\n: LTD BLOCK 5B (PRITECH I!)BELLANOUR\n\nVILLAGE VARTHUR HOBLI
BANGALORE\n560066\n\n: RAPID ELEGANCE+1.6 TOICR MT\n\n: SALOON NO.OF
CYL 34\nUNLADEN WT ; 1205\n: 06/2014 SEATING 18\n: DIESEL STOG/SLPR
\n29/08/2029 cc 1998 . —\n: LTT Registering Authority\n
\nBENGALURU(E)'
```

Fig 3. Tesseract extracted text

4. **Regex**

- **Regular expressions** are implemented to uniquely identify every value for a given key-value pair, since the RC cards of the various states in India have different syntaxes for their keys.

- Each piece of information was identified to have a particular structure to it, on the basis of which, these unique expressions were generated.



Components [edit]

Modern VINs are based on two related standards, originally issued by the International Organization for Standardization (ISO) in 1979 and 1980: ISO 3779[4] and ISO 3780,[5] respectively. Compatible but different implementations of these ISO standards have been adopted by the European Union and the United States, respectively.[6]

The VIN comprises the following sections:

Standard	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
ISO 3779	World manufacturer identifier			VDS						VIS							
European Union[7] more than 500 vehicles/year	World manufacturer identifier			Indication of *the general characteristics of the vehicle*						Indication that provides *clear identification of a particular vehicle*							
European Union[7] 500 or fewer vehicles/year	World manufacturer identifier		9	Indication of *the general characteristics of the vehicle*						Indication that provides *clear identification of a particular vehicle*							
North America more than 2000 vehicles/year	World manufacturer identifier			Vehicle attributes					Check digit	Model year	Plant code	Sequential number					
North America 2000 or fewer vehicles/year	World manufacturer identifier		9	Vehicle attributes					Check digit	Model year	Plant code	Manufacturer identifier		Sequential number			

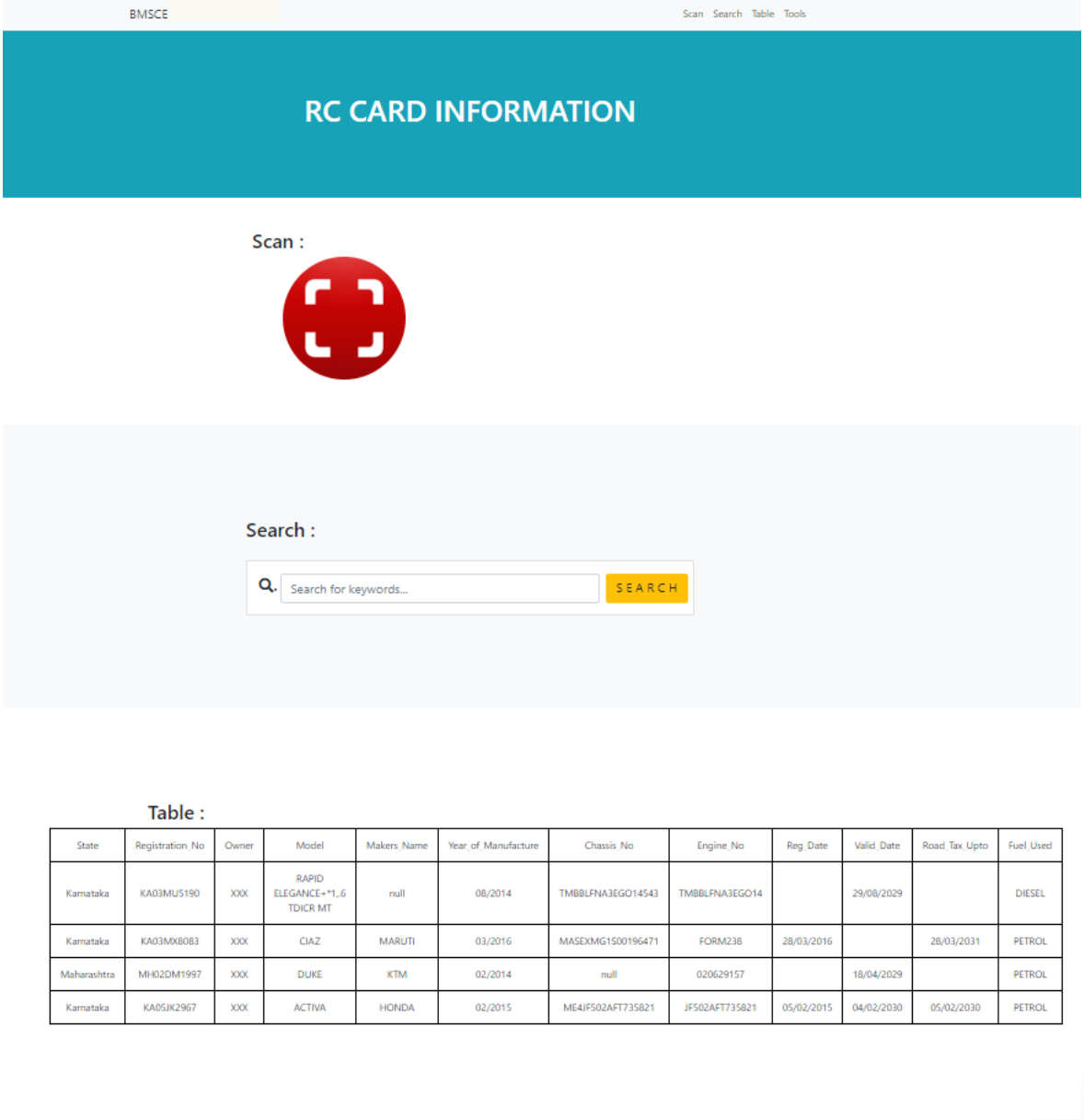
Fig 4. Chassis no. structure

```
120
121 REGNO = re.compile(r'[A-Z]{2}([0-9]{1,2}|O[0-9]?)[A-Z]{1,2}[0-9]{3,4}\s')
122 CHASSIS = re.compile(r'[0-9A-Z]{12}[0-9]{5}\s')
123 ENGINE = re.compile(r'[0-9A-Z]{6}[0-9A-Z]{1,8}\s')
124 FUEL = re.compile(r'PETR(O|0)L')
125 MAKERS = ['MARUTI','TOYOTA','HYUNDAI','CHEVORLET','TATA','SKODA','HONDA','KTM',
126 DATE = re.compile(r'[0-9]{2}/[0-9]{2}/[0-9]{4}')
127 MANUDATE = re.compile(r'[0-9]{2}(/|-)[0-9]{4}')
128
129 STATES = {'TN':'Tamil Nadu',
130           'AN':'Andaman and Nicobar',
131           'AP':'ANDHRA PRADESH',
132           'KA': 'Karnataka',
133           'HR': 'Haryana',
134           'TS': 'Telangana',
135           'BR': 'Bihar',
136           'MP': 'Madhya Pradesh',
137           'MH':'Maharashtra',
138          }
```

Fig 5. Example Regex

❖ Web Application

- Web Application was designed using Django, HTML, JavaScript and CSS.
- Interface included a administrator view of the database of the users with optional search querying functionality.



State	Registration_No	Owner	Model	Makers_Name	Year_of_Manufacture	Chassis_No	Engine_No	Reg_Date	Valid_Date	Road_Tax_Upto	Fuel_Used
Karnataka	KA03MU5190	XXX	RAPID ELEGANCE+*1,,6 TDICR MT	null	08/2014	TMBBLFNA3EGO14543	TMBBLFNA3EGO14		29/08/2029		DIESEL
Karnataka	KA03MX8083	XXX	CIAZ	MARUTI	03/2016	MASEXMG1S00196471	FORM238	28/03/2016		28/03/2031	PETROL
Maharashtra	MH02DM1997	XXX	DUKE	KTM	02/2014	null	020629157		18/04/2029		PETROL
Karnataka	KA05JK2967	XXX	ACTIVA	HONDA	02/2015	ME4JF502AFT735821	JF502AFT735821	05/02/2015	04/02/2030	05/02/2030	PETROL

Fig 6. Web Application

Output

- Images were extracted, processed, filtered and finally read from into a structured format and then inserted into the **SQLite Database.**
- Below is a view of the Database of stored RC card information for the admin of the company.

Table :

State	Registration_No	Owner	Model	Makers_Name	Year_of_Manufacture	Chassis_No	Engine_No	Reg_Date	Valid_Date	Road_Tax_Upto	Fuel_Used
Karnataka	KA03MU5190	XXX	RAPID ELEGANCE+*1,.6 TDICR MT	null	08/2014	TMBBLFNA3EGO14543	TMBBLFNA3EGO14		29/08/2029		DIESEL
Karnataka	KA03MX8083	XXX	CIAZ	MARUTI	03/2016	MASEXMG1S00196471	FORM238	28/03/2016		28/03/2031	PETROL
Maharashtra	MH02DM1997	XXX	DUKE	KTM	02/2014	null	020629157		18/04/2029		PETROL
Karnataka	KA05JK2967	XXX	ACTIVA	HONDA	02/2015	ME4JF502AFT735821	JF502AFT735821	05/02/2015	04/02/2030	05/02/2030	PETROL

Fig 6. Web Application

Conclusion

The findings from this project allow us to successful assure a insurance based company of accurate, automated data retrieval from the aforementioned RC card.

Not restricted to RC cards, this scanning module in its entirety, can be considered generic to all sorts of card scanning systems. This provides for further various forms of implementation of the same base technologies.

References

1. https://pypi.org/project/pytesseract/
2. https://www.pyimagesearch.com/2017/07/10/using-tesseract-ocr-python/
3. https://cv-tricks.com/opencv-dnn/edge-detection-hed/
4. https://docs.djangoproject.com/en/1.8/intro/tutorial01/
5. https://www.tutorialspoint.com/android/
6. https://developer.android.com/reference/java/net/Socket

Technologies Used