PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE - 411043
Department of Computer Engineering
S.No.-27, Pune Satara Road, Dhankawadi, Pune-411043

Project Based Learning-II

Project Report

Batch: H1 Date: 31st May, 2021

VEHICLE IDENTIFICATION & LICENSE PLATE DETECTION SYSTEM

Under the Guidance of –

Prof. Virendra Bagade

GROUP MEMBERS:

Soham Rakhunde	21168
Gauri Takawane	21181
Viraj More	21184
Brijesh Yerram	21186

Index

Table of Contents:

INDEX

Table of Pictures

(Comes under Output Screenshots)

INTRODUCTION

MOTIVATION:

The escalating increase of contemporary urban and national road networks over the last decades emerged the need of efficient monitoring and management of road traffic. Meanwhile, rising vehicle use causes social problems such as accidents, traffic congestion, and consequent traffic pollution. Automatic Number Plate Recognition has been one of the useful approaches for vehicle surveillance. It can be applied at number of public places for fulfilling some of the purposes like traffic safety enforcement, automatic toll text collection, car park system and Automatic vehicle parking system. In automated systems, people utilize computer-based expert systems to analyse and handle real-life problems such as intelligent transportation systems. Presently number plate detection and recognition processing time is less than 50 milliseconds in many systems.

OBJECTIVE:

To identify the license plate of vehicle and scrape the details of the vehicle which is registered under this license plate.

PURPOSE:

The escalating increase of contemporary urban and national road networks over the last decades emerged the need of efficient monitoring and management of road traffic. Meanwhile, rising vehicle use causes social problems such as accidents, traffic congestion, and consequent traffic pollution. Automatic Number Plate Recognition has been one of the useful approaches for vehicle surveillance. It can be applied at number of public places for fulfilling some of the purposes like traffic safety enforcement, automatic toll text collection, car park system and Automatic vehicle parking system. In automated systems, people utilize computer-based expert systems to analyze and handle real-life problems such as intelligent transportation systems. Presently number plate detection and recognition processing time are less than 50 milliseconds in many systems.

INTENDED AUDIENCE:

In this system, vehicles are identified or recognized using their number plate or license plate. It uses image processing techniques to extract the vehicle number plate from digital images. Such systems usually comprise of two components. A camera that used in capturing of vehicle number plate images, and software that extracts the number plates from the captured images by using a character recognition tool that allows for pixels to be translated into numerical readable characters. It is used widely in various fields such as:

- Automated 24*7 human free Surveillance system
- High Security Institutions like Military/Government area
- Apartments/ Businesses for automated logging of entry-exit of Vehicles
- Automatic payment of tolls on highways or bridges
- Parking management systems
- Vehicle tracking & Traffic monitoring
- Surveillance systems

OVERALL DESCRIPTION

a. FUNCTIONAL REQUIREMENT

(Following are the different classes made for the project with each having an individual purpose and a set of defined methods which improve the functionality.)

1. *App:*

1.1. Inherits ttk.ThemedTk and acts as Tkinter main class and implements the UI/UX entry point. This class outputs a Window on which we can display application UI elements.

1.2. *Functions:*

1.2.1. **side_bar():** Creates a side Menu in a frame which is then displayed onto the App window.

1.2.2. **switch_frame():** Implements a Frame Handler that is called when one of the menu options is selected and shows the selected frame (HomePage or historyPage) on the app window.

2. *HomePage:*

2.1. Inherits tkinter.frame class and implements the Home page UI of app that displays a button to import a video for Processing or the Video output of the execution of the program and the extracted License plate.

2.2. *Functions:*

2.2.1. **open_vid():** After pressing the button to open a video this call-back function is called by the button press event. To improve the execution speed of the program since opening a video is IO bound process that means CPU has to halt until it fetches the video. To avoid this wastage, we spawn a Thread that makes the CPU do further processing while the program Concurrently fetches the video. This function call asks User for video directory via file explorer. This directory path is further sent to image refresh function in a new Thread.

2.2.2. **image_refresh ():** The directory path is now further sent to the instance of class VideoProcessor. We use this object to refresh the Images displayed on the homepage frame by frame. As soon as VideoProcessor processes a new frame Concurrently with the use of threads we can refresh the Video framewise on the Homepage. This helps us see the Video being processed in real-time. Same goes for the last recognised License plate image which is also displayed on the homepage.

2.2.3. **updater():** This method refreshes the clock on homepage every second. This clock is used to simulate the camera/video capture time. As this time will be saved with the processed vehicles details as entry time.

3. *Video Processor*

3.1. Processes the video, frame by frame and searches for the contours that resemble the number plate of a car and crops it. This image is then stored into the database and is also sent for further processing which is segmenting the characters.

3.2. *Functions:*

3.2.1. **VideoCapture()** (Constructor)**:** The video is loaded and broken down into frames using VideoCapture() function of cv2 in the constructor. Similar frames are discarded (To avoid Repetition) and then these frames are sent to the processFramewise() function.

3.2.2. **processFramewise():** This function accepts the frame as an input and outputs the cropped license plate image. This is achieved by using some of the functions/modules of the opencv library. Some of them are listed below-:

3.2.2.1. cv2.gray()

3.2.2.2. cv2.drawcontours()

3.2.2.3. cv2.boundingrect()

3.2.2.4. cv2.Canny()

After cropping the license plate image. The frame and this cropped image are loaded into the database and then the control shifts to the next part which is segmenting the characters.

4. *Recognition:*

4.1. This class focuses on realizing the input license plate and identification of its characters.

4.2. *Functions:*

4.2.1. **showOriginal():** Used to display and realize the input image(license plate) from the previous class.

4.2.2. **process_Image():** This function is the main function resposible for identifying the required components. In this we resize the license plate image and convert it into gray along with thresholding to seperate each and every component for further processing. Further we apply connected component analysis on this thresholded image to get information about the dimensions of each and every component along with its centroid. Then we loop over this component list and select particular components which satisfy all the conditions of required dimensions. Finally we mask these selected components on an empty image for applying character recognition on it.

4.2.2.1. **cv2.resize():** OpenCV comes with a function cv2.resize() for scalling purpose. The size of the image can be specified manually, or you can specify the scaling factor.

Different interpolation methods are used. Preferable interpolation methods are cv.INTER_AREA for shrinking and cv.INTER_CUBIC (slow) & cv.INTER_LINEAR for zooming. By default, the interpolation method cv.INTER_LINEAR is used for all resizing purposes.

4.2.2.2. **cv2.cvtColor():** The function converts an input image from one color space to another. In case of a transformation to-from RGB color space, the order of the channels should be specified explicitly (RGB or BGR). Note that the default color format in OpenCV is often referred to as RGB but it is actually BGR (the bytes are reversed). So the first byte in a standard (24-bit) color image will be an 8-bit Blue component, the second byte will be Green, and the third byte will be Red. The fourth, fifth, and sixth bytes would then be the second pixel (Blue, then Green, then Red), and so on.

4.2.2.3. **cv2.threshold():** The function applies fixed-level thresholding to a multiple-channel array. The function is typically used to get a bi-level (binary) image out of a grayscale image or for removing a noise, that is, filtering out pixels with too small or too large values. There are several types of thresholding supported by the function. They are determined by type parameter. Also, the special values THRESH_OTSU or THRESH_TRIANGLE may be combined with one of the above values. In these cases, the function determines the optimal threshold value using the Otsu's or Triangle algorithm and uses it instead of the specified thresh.

4.2.2.4. **cv2.connectedComponentsWithStats():** This function computes the connected components labeled image of boolean image and also produces a statistics output for each label image with 4 or 8 way connectivity - returns N, the total number of labels [0, N-1] where 0 represents the background label. ltype specifies the output label image type, an important consideration based on the total number of labels or alternatively the total number of pixels in the source image.

4.2.2.5. **cv2.rectangle():** The function cv::rectangle draws a rectangle outline or a filled rectangle whose two opposite corners are given.

4.2.2.6. **cv2.circle():** The function cv::circle draws a simple or filled circle with a given center and radius.

4.2.2.7. **detectLicense():** In this function we take the input as resultant masked image of process_Image() function and apply the optical character recognition on it. We apply the pytesseract.image_to_string() on image and it detects and returns the license plate no. in form of string.

4.2.2.8. **pytesseract.image_to_string(res):** The image_to_string() method converts the image text into a Python string which you can then use however you want.

5. *HistoryPage:*

5.1. Inherits tkinter.frame class and implements the History page UI that displays all the Sqlite Database Entries on the GUI using tkinter.ttk.treeview.

5.2. This class also makes connection with the database and also initialises the cursor to the database as the class member variables.

5.3. We can also use the Search bar on this page that enables us to search for list of entries which match the query from the database on the basis of Vehicle Registration number (License plate number), date, time, and Address.

5.4. This page also has a export/download bar that helps the user to get All the Entries / Searched Entries / Selected Entries which can be used to download these entries as an Excel document seamlessly or download the images of number Plate/whole frame from the database directly for further usages like Machine Learning datasets.

5.5. Functions:

5.5.1. **searchBar():** Implements the UI for search bar with information labels, a combo box for selecting the type of query, one entry box to enter the query and two Buttons one for Searching and other one to Reset the search results. The Search bar enables us to search for list of entries which matches the query from the database on the basis of Vehicle Registration number (License plate number), date, time, and Address. This result is shown by the ttk.treeview object of this Class/Page.

5.5.2. **dbTable():** It builds the ttk.treeview on this Page including a vertical scrollbar with synchronised state. This function adds columns and respective names and sizes. But the Table remains empty.

5.5.3. **downloadBar():** Implements the UI for Export/Download bar with information labels, two combo boxes for selecting the type of query and the type of export data, and one Button to download it.
Download bar enables user to export the saved photo of the vehicle, its license plate or the data as Excel document of the selected entry / searched entries / all entries from the database.

5.5.4. **onSingleClickEntry():** Call back function for Entry box that erases its default message when user starts entering Input that is query.

5.5.5. **onDoubleClick():** Call back function for Treeview table row(entry) which on double click opens a tkinter. TopLevel window and destroys the main window of the program.

5.5.6. **updateTable():** This function erases the contents of the treeview table. And adds the new contents from a list retrieved from the database query. Also, it adds the odd-even style to the rows.

5.5.7. **searchTree():** Callback function to the Search button. This function retrieves the query terms from the combo box and entry box of the search bar. It then executes the query on the database cursor and the resulting list is sent to updateTable() to change the contents of the treeview.

5.5.8. **download():** Callback function to the Download button. This function retrieves the information from the two combo boxes about the download type. Then asks the user for the save directory with file explorer. Then according to the request, we retrieve the list from database and for each image/data we save it in the directory. For writing the excel file we use XLWT package.

6. *DetailsPage:*

6.1. Inherits tkinter.TopLevel class and implements the Details page UI that displays all the information about particular entry on the GUI. We display the License plate, address, visit number, total visits, last seen time, PUC/Insurance/Registration validity. It also displays the photo of the Vehicle taken during processing and the number plate photo.

6.2. And finally, we have a treeview table which shows all the visits of this particular vehicle with its date/ time, visit number and id. These entries can be accessed directly from here by using double click which will open the instance of this class in a new window. After closing this window, we return back to the main window.

6.3. Functions:

6.3.1. **textInfoFrame():** Implements the UI for String data and both of the photos on the window. This includes changing the size of the Photos to fit correctly retaining the Aspect Ration and also these images are in binary BLOB format of sqlite3 which is converted into an PIL image. It also uses methods from error codes class to decode the binary code into a String of readable data for the UI.

6.3.2. **treeTable():** It builds the ttk.treeview on this Page including a vertical scrollbar. This function adds columns and respective names and sizes. It also adds the new content to the treeview table from a list retrieved from the database query in the constructor. Also, it adds the odd-even style to the rows.

6.3.3. **onDoubleClick():** Callback function for Treeview table row(entry) which on double click opens a tkinter.TopLevel window and destroys the current TopLevel window of the program.

6.3.4. **closeHandler():** This is an overloaded method and changes the behaviour of toplevel window being closed by the user. It destroys itself but before that it calls its master to be visible again that is the Main window to be visible.

7. *HoverButton():*

7.1. It inherits tkinter.Button class and changes its behaviour to change the colour of a button when mouse hovers over it. It binds to functions to the event. And when the event occurs the callback functions are executed. This allows us to make a better UI as well as responsive application.

7.2. *Functions:*

7.2.1. **on_enter(event):** Changes the background colour of this button when cursor starts hovering. Called by <Enter> event.

7.2.2. **on_leave(event):** Changes the background colour back to the original colour of this button when cursor stops hovering over it. Called by <Leave> event.

8. *DataClass:*

8.1. This class is used to store all the data about a single vehicle to make the data easily maintainable and readable. The data members are:

8.1.1. numPlate – String

8.1.2. timestamp – String from dateTime object

8.1.3. vehicleName – String

8.1.4. address – String

8.1.5. errCode – int encoded by errorcodes class

8.1.6. photo – Binary format of image

8.1.7. plateImage – Binary format of number plate image

9. *Scraper:*

9.1. Inherits Rekognition class from which we set an instance of DataClass that is further used in this class retaining its state. This class uses Selenium webdriver(which uses edge browser) for web scraping/crawling the website called “Vahan.nic.in” from which we get the data of the vehicle whose number plate is extracted and saved DataClass instance as ‘numPlate’.

9.2. The program first logs in the website and fills the field of vehicle registration number. But to avoid a program to use this feature they have used ‘captcha’ to ensure only humans pass through. To overcome the problem, we capture this captcha and use OpenCV to remove noise and pass it to Tesseract OCR that recognizes the characters after getting these characters we proceed to fill it in the website.

9.3. If the captcha is correct, we proceed to the information page for the vehicle else we have to try another captcha. After we get the information, we send all this information to the database insert functions for inserting it into the database successfully.

9.4. *Functions:*

9.4.1. **login():** Uses Selenium to fill in the username and password to the website and waits until the webdriver successfully logs into the site.

9.4.2. **captchaSolve():** This function waits until we load the captcha image. Once it loads we take the screenshot and pass it to tesseract_captcha_ocr function which returns the captcha string of after that the webdriver fills this captcha text into website and presses the Vahan Search button. If it fails, we get a new Captcha which it repeats until it gets to the next page.

9.4.3. **finalScrape():** After we reach final page we extract info like vehicleName, address, regDate, PUCdate and insuaranceDate we save this info in DataClass instance and the dates are sent to saveInDb() function.

9.4.4. **send_discrete_keys():** The website is designed to avoid automated programs due to which the program needs to write one letter at a time in the entry box of the website.

9.4.5. **saveInDb():** The function calls the static method of errorcodes class encode () which takes the three dates converts it into datetime objects and compare it to make an int code from these three dates.

Then the photos stored in the DataClass objects are converted into binary format and sent into database.inset() function as a parameter.

9.4.6. **tesseract_captcha_ocr():** Converts the captcha image into a numpy array so that its usable with OpenCV and finally it is converted into OpenCV image.

OpenCV is used to remove all the noise in image by turning it into black and white, Thresholding the image with OTSU and binary thresholding, and Gaussian Blur.

This image is now sent to tesseract OCR using pytesseract library which returns a string. This string is returned by this function.

9.4.7. **is_license_format():** Before scraping we check if the previous module has a valid license plate recognition if not then returns false. This avoids the issue of invalid license plate being scarped to find no vehicle found error.

9.4.8. **license_plate_parser():** Due to similarities between 0(zero) and O(letter) Tesseract OCR can sometimes output wrong values that is where a letter O is present it could recognise number zero. This function checks the string and replaces such errors.

10. errorcodes:

10.1. Class to encode and decode error codes.

10.2. Functions:

*10.2.1.***encode():** Static method which takes parameters as registration date, PUC date and insurance data converts it into datetime instance and compares it with the current time to see if any date exceeds validity. If it does we use bitwise OR to make a 3 bit code representing each date if its validity is expired. This code is useful as we do not have to store 3 dates reducing the space occupied in the database.

*10.2.2.***decode():** Static method which takes parameters as errorcodes and return a sting of the format which date exceeds validity. This function is used to decode the error code value and give a readable meaning out of the code.

B. OPERATING ENVIRONMENT

(Database, S/W and H/W requirement)

1. DATABASE:

1.1. This class implements sqlite3 related functions for database management. The columns are as follows with the data type:

1.1.1. numPlate – text

1.1.2. timeStamp – text

1.1.3. name – text

1.1.4. address - text

1.1.5. errCode – integer

1.1.6. photo - blob

1.1.7. platePic – blob

1.2. *Functions:*

1.2.1. **Create():** Makes a connection to the database and creates a cursor. Creates a new empty table in this SQLite database with the given column names. Finally commits the cursor and closes the database.

1.2.2. **Insert():** Makes a connection to the database and creates a cursor. Executes the insert query and uses the parameters to insert into the database and the binary format photos are further converted by SQlite3.binary() to create BLOB format compatible binary photos. Finally commits the cursor and closes the database.

2. SOFTWARE REQUIREMENTS:

2.1. Python

2.1.1. Python is an interpreted, high-level, general-purpose programming language. It's design philosophy emphasizes code readability through use of significant whitespace.

2.1.2. Funtional Requirement

2.1.2.1. Python: any version above 3.7

2.2. Opencv

2.2.1. OpenCV provides a real-time optimized Computer Vision library and tools.

2.2.2. Funtional Requirement

2.2.2.1. OpenCV – v4.5.1.48:

2.2.2.2. PIL – to open an Image, crop it,etc.

2.3. Selenium

2.3.1. Selenium is an open source automation tool which can used to automate browser and also for crawling.

2.3.2. Functional Requirements:

2.3.2.1. Selenium – v3.141.0

2.4. Tesseract:

2.4.1. TESSERACT is an OCR engine with support for Unicode and the ability to recognize more than 100 languages out of the box. It can be trained to recognize other languages as well.

2.4.2. Functional Requirements:

2.4.2.1. pyTesseract – v0.3.7

2.4.2.2. Tesseract OCR: Library1: pytesseract – python-wrapper

2.5. Microsoft Edge Browser:

2.5.1. Microsoft Edge is a cross-platform web browser developed by Microsoft.

2.5.2. Functional Requirements:

2.5.2.1. EdgeOptions()

2.5.2.2. Webdriver.edge()

3. HARDWARE REQUIREMENTS:

3.1. Camera/Pre-recorded Video:

3.1.1. The software can work with two types of inputs that are as follows:

3.1.1.1. Live Video Feed from camera.

3.1.1.2. Pre-recorded video footage.

3.1.1.3. The program is compatible with Windows OS, Linux OS and Mac OS. For prerecorded video footage H.264 encoded DIVX and mp4 formats are supported.

3.1.2. The following are the requirements for the traffic enforcement camera:

3.1.2.1. Lens: 3.6mm fixed lens with high resolution video up to 1920x1080

3.1.2.2. Wi-fi: supports both public and private networks.

3.1.2.3. LED: Built-in infrared LED lights with crystal clear image quality

3.1.2.4. Night-Vision: It should adopt ICR (Infrared Filter) technology which supports switch between day and night mode automatically.

3.2. Computer:

3.2.1. The following are the requirements of the Computer:

(Any average i3 processor laptop/PC will do. The details listed below are subjected to vary.)

3.2.1.1. Processor: Minimum Quad core CPU(atleast 6 threads).

3.2.1.2. GPU: Not Required

3.2.1.3. Memory: Minimum 4GB RAM

3.2.1.4. Storage: Minimum 1TB RAID – 1 HDD Storage with redundancy and backup.

3.2.1.5. Display: Generic HD monitor.

3.2.1.6. Operating Systems: Linux / Windows / MacOS

FLOWCHART

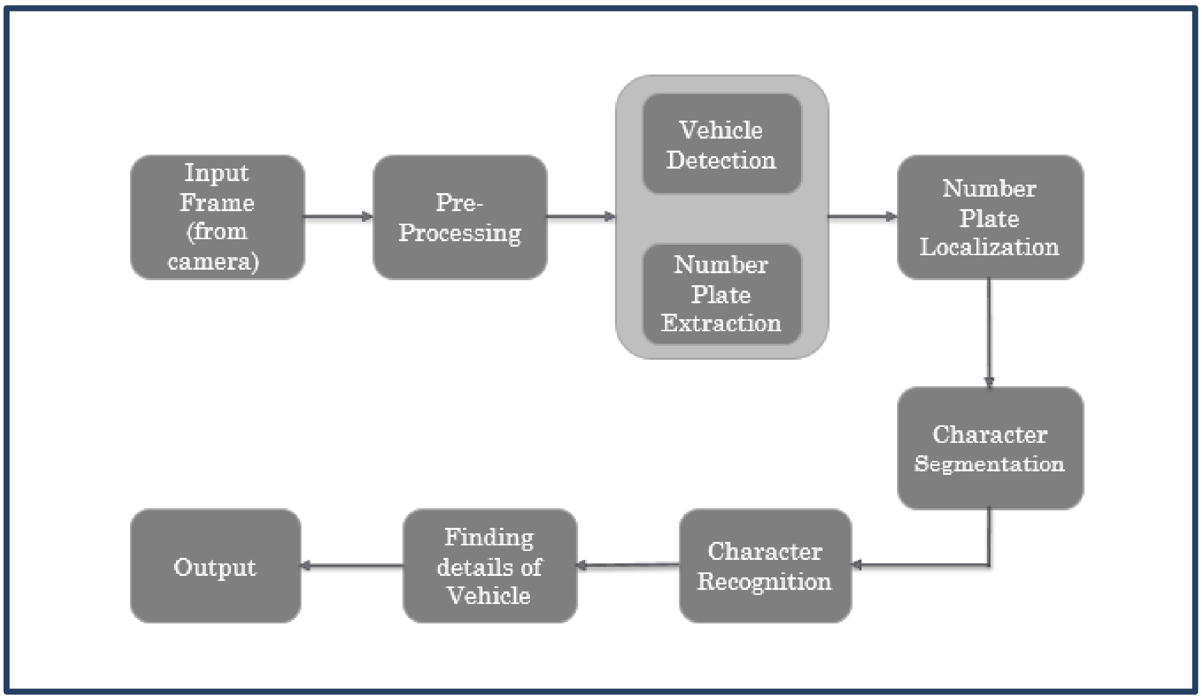


Figure 0: Flowchart of the whole process.

IMPLEMENTATION DETAILS ALONG WITH SCREENSHOTS:

1. IMPLEMENTATION DETAILS- CODE:

1.1. App:

Inherits ttk.ThemedTk and acts as Tkinter main class and implements the UI/UX entry point. This class outputs a Window on which we can display application UI elements.

Functions:

- side_bar()
- switch_frame()

1.2. HomePage:

Inherits tkinter.frame class and implements the Home page UI of app that displays a button to import a video for Processing or the Video output of the execution of the program and the extracted License plate.

Functions:

- open_vid()
- image_refresh ()
- updater()

1.3. Video Processor

Processes the video, frame by frame and searches for the contours that resemble the number plate of a car and crops it. This image is then stored into the database and is also sent for further processing which is segmenting the characters.

Functions:

- VideoCapture() (Constructor)
- processFramewise()

After cropping the license plate image. The frame and this cropped image are loaded into the database and then the control shifts to the next part which is segmenting the characters.

1.4. Rekognition:

Responsible for realizing the license plate image and identifying the important components.

Functions:

- showOriginal()
- process_Image()

1.5. HistoryPage:

1.5.1.1. Inherits tkinter.frame class and implements the History page UI that displays all the Sqlite Database Entries on the GUI using tkinter.ttk.treeview.

1.5.1.2. This class also makes connection with the database and also initialises the cursor to the database as the class member variables.

1.5.1.3. We can also use the Search bar on this page that enables us to search for list of entries which match the query from the database on the basis of Vehicle Registration number (License plate number), date, time, and Address.

1.5.1.4. This page also has a export/download bar that helps the user to get All the Entries / Searched Entries / Selected Entries which can be used to download these entries as an Excel document seamlessly or download the images of number Plate/whole frame from the database directly for further usages like Machine Learning datasets.

Functions:

- searchBar()
- dbTable()
- downloadBar()
- onSingleClickEntry()
- onDoubleClick()
- updateTable()
- searchTree()
- download()

1.6.DetailsPage:

1.6.1.1. Inherits tkinter.TopLevel class and implements the Details page UI that displays all the information about particular entry on the GUI. We display the License plate, address, visit number, total visits, last seen time, PUC/Insurance/Registration validity. It also displays the photo of the Vehicle taken during processing and the number plate photo.

1.6.1.2. And finally, we have a treeview table which shows all the visits of this particular vehicle with its date/ time, visit number and id. These entries can be accessed directly from here by using double click which will open the instance of this class in a new window. After closing this window, we return back to the main window.

Functions:

- textInfoFrame()
- treeTable()
- onDoubleClick()
- closeHandler()

1.7.HoverButton():

It inherits tkinter.Button class and changes its behaviour to change the colour of a button when mouse hovers over it. It binds to functions to the event. And when the event occurs the callback functions are executed. This allows us to make a better UI as well as responsive application.

Functions:

- on_enter(event)
- on_leave(event)

1.8.DataClass:

This class is used to store all the data about a single vehicle to make the data easily maintainable and readable. The data members are:

- *numPlate – String*
- *timestamp – String from dateTime object*
- *vehicleName – String*

- *address – String*
- *errCode – int encoded by errorcodes class*
- *photo – Binary format of image*
- *plateImage – Binary format of number plate image*

1.9. Scraper:

1.9.1. Inherits Rekognition class from which we set an instance of DataClass that is further used in this class retaining its state. This class uses Selenium webdriver(which uses edge browser) for web scraping/crawling the website called "Vahan.nic.in" from which we get the data of the vehicle whose number plate is extracted and saved DataClass instance as 'numPlate'.

1.9.2. The program first logs in the website and fills the field of vehicle registration number. But to avoid a program to use this feature they have used 'captcha' to ensure only humans pass through. To overcome the problem, we capture this captcha and use OpenCV to remove noise and pass it to Tesseract OCR that recognizes the characters after getting these characters we proceed to fill it in the website.

1.9.3. If the captcha is correct, we proceed to the information page for the vehicle else we have to try another captcha. After we get the information, we send all this information to the database insert functions for inserting it into the database successfully.

Functions:

- login()
- captchaSolve()
- finalScrape()
- send_discrete_keys()
- saveInDb()
- tesseract_captcha_ocr()
- is_license_format()
- license_plate_parser()

1.10. *Errorcodes:*

Class to encode and decode error codes.

Functions:

- encode()
- decode()

1.11. DataBase:

This class implements sqlite3 related functions for database management. The columns are as follows with the data type:

- *numPlate – text*
- *timeStamp – text*
- *name – text*
- *address - text*
- *errCode – integer*
- *photo - blob*
- *platePic – blob*

Functions:

- Create()
- Insert()

2. OUTPUT SCREENSHOTS:

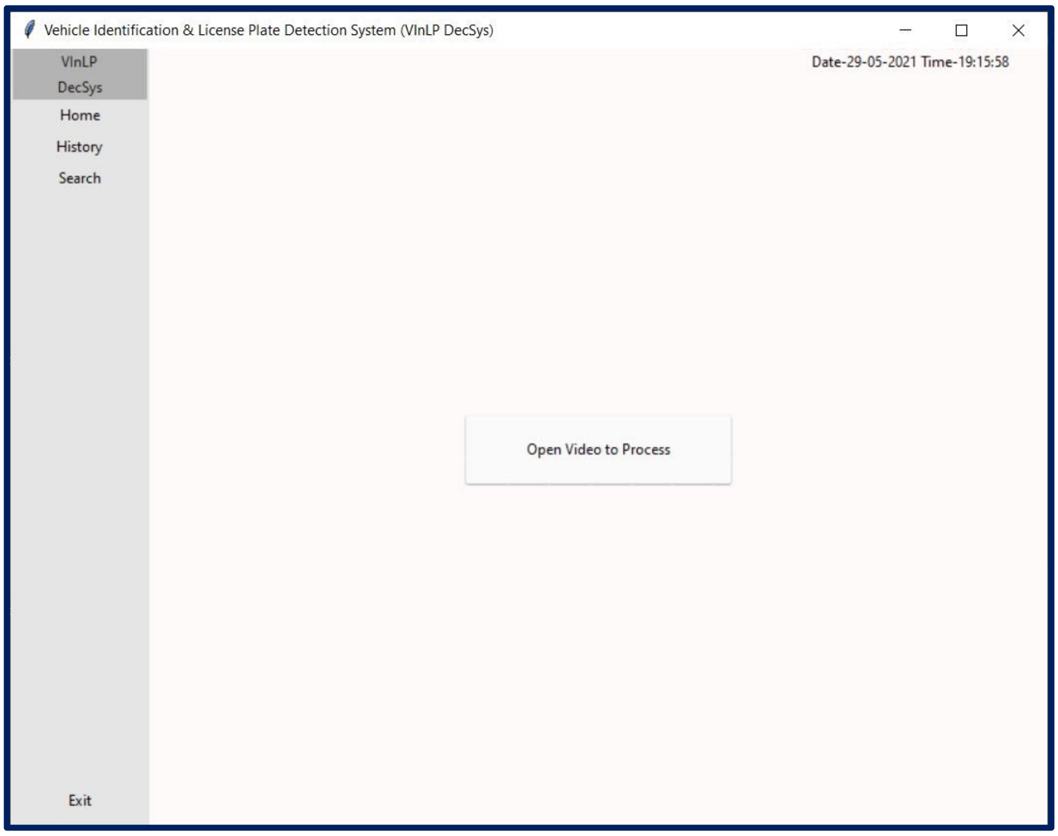

Vehicle Identification & License Plate Detection System (VInLP DecSys)
VInLP
DecSys
Home
History
Search
Exit
Date-29-05-2021 Time-19:15:58
Open Video to Process

Figure 1: Tkinter Main window – home screen with option to process a video.

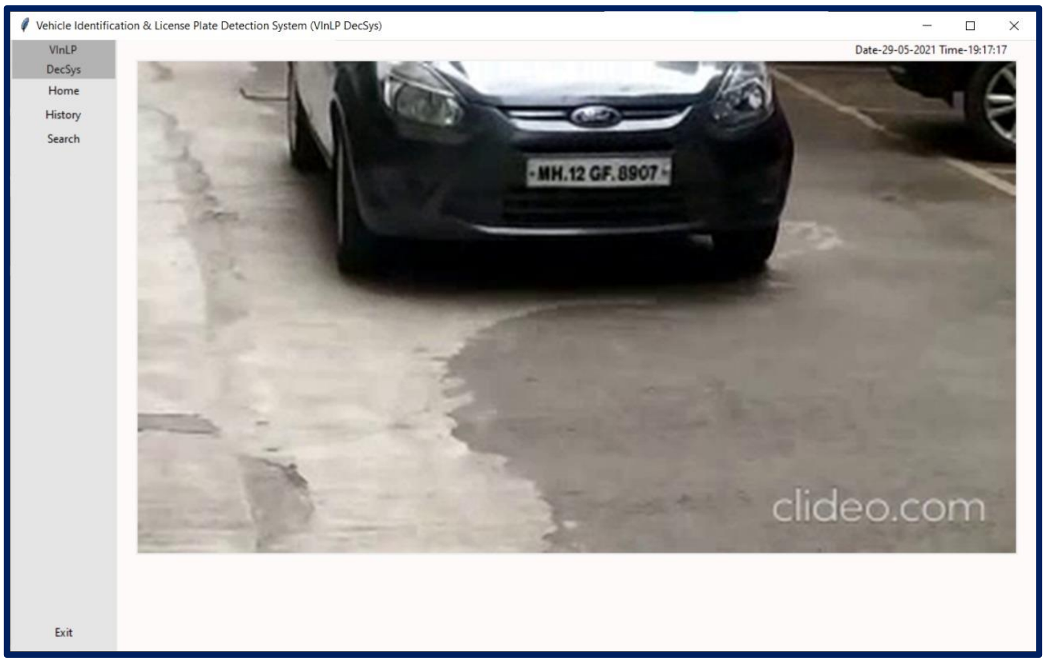

Vehicle Identification & License Plate Detection System (VInLP DecSys)
VInLP
DecSys
Home
History
Search
Exit
Date-29-05-2021 Time-19:17:17
MH.12 GF.8907
clideo.com

Figure 2:Video under processing and Simultaneous Video stream on the UI.

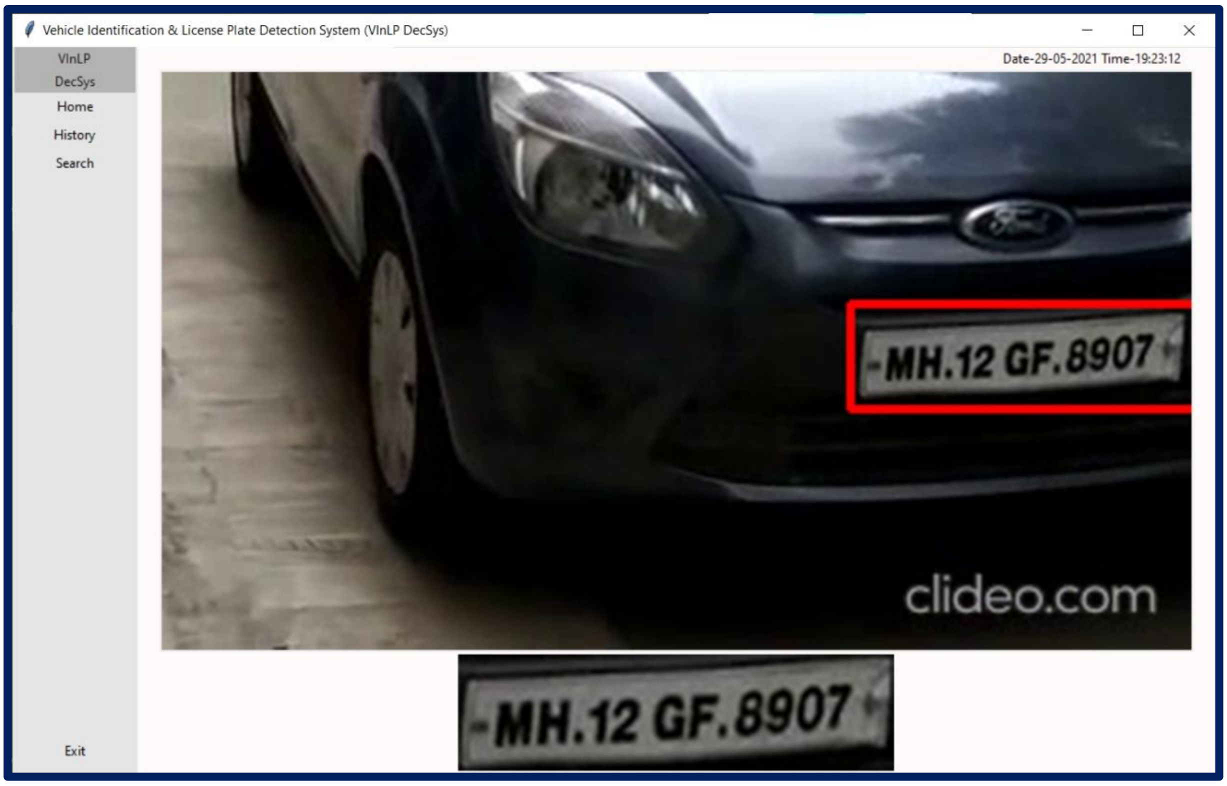

Vehicle Identification & License Plate Detection System (VInLP DecSys)
VInLP DecSys
Home
History
Search
Exit
Date-29-05-2021 Time-19:23:12
MH.12 GF.8907
clideo.com
MH.12 GF.8907

Figure 3: a: Displaying on UI - Detected Number Plate.

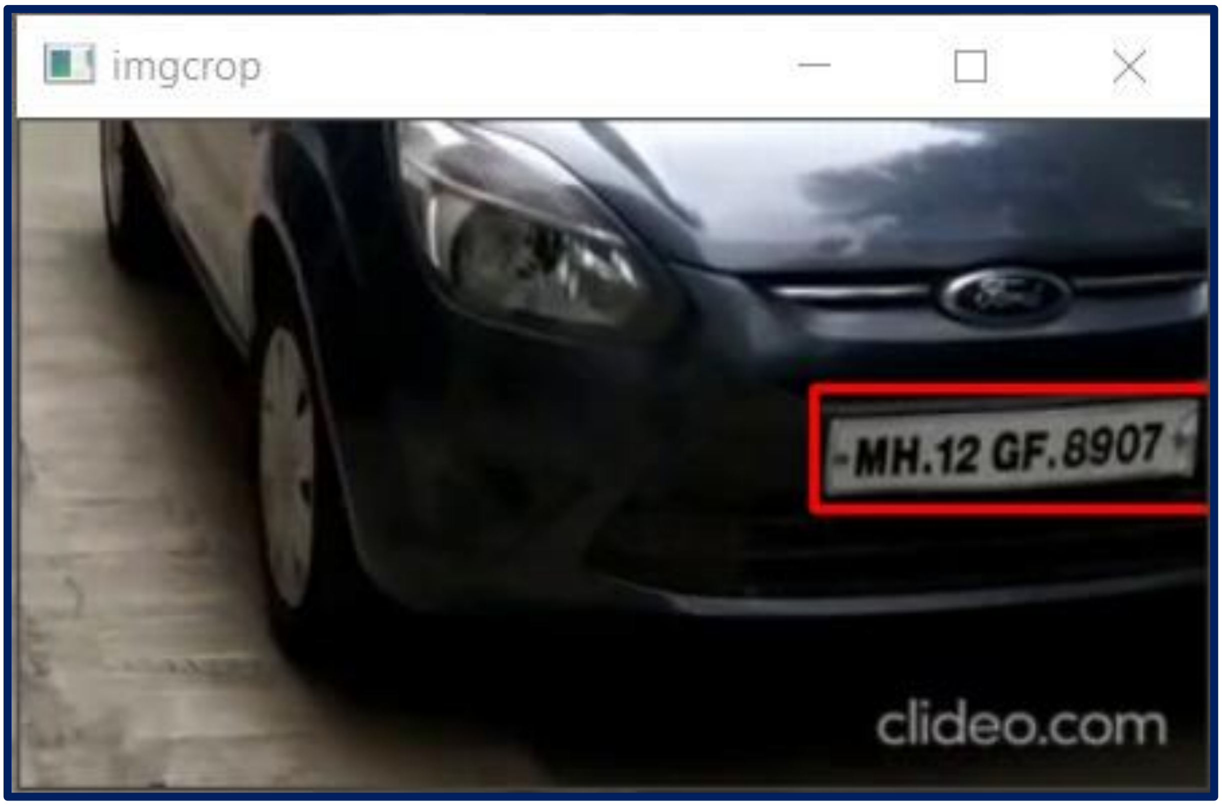


Figure 3: b: OpenCV displaying output as - Number plate detected.

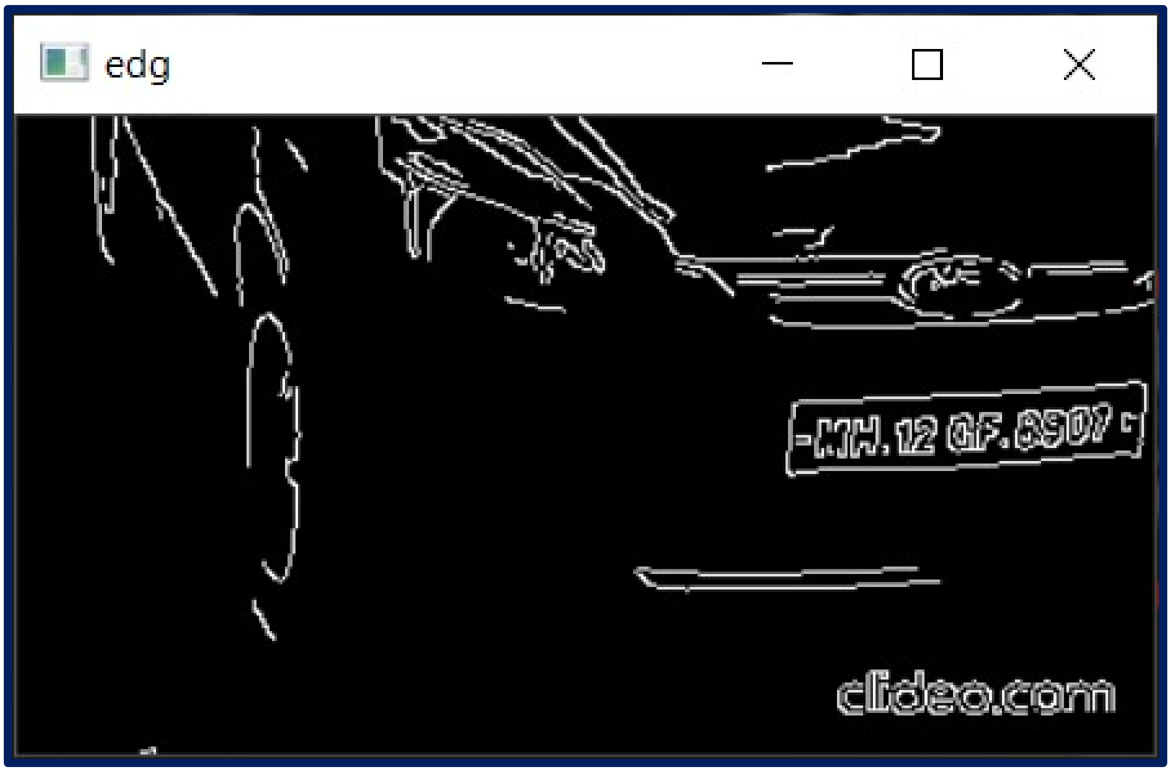


Figure 4: OpenCV - Noise free Edge Image.

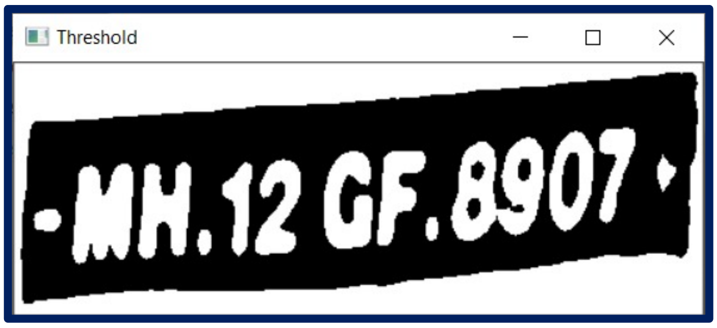


Figure 513: OpenCV extracted Number plate.

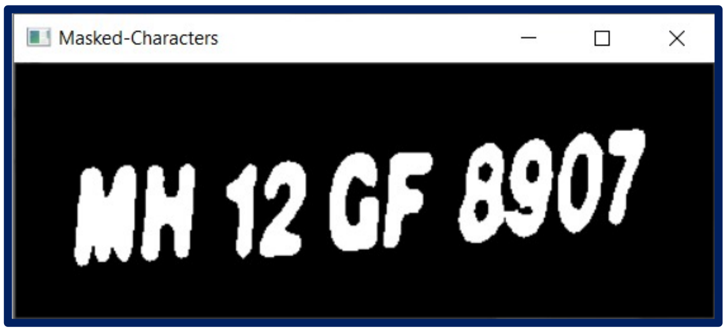
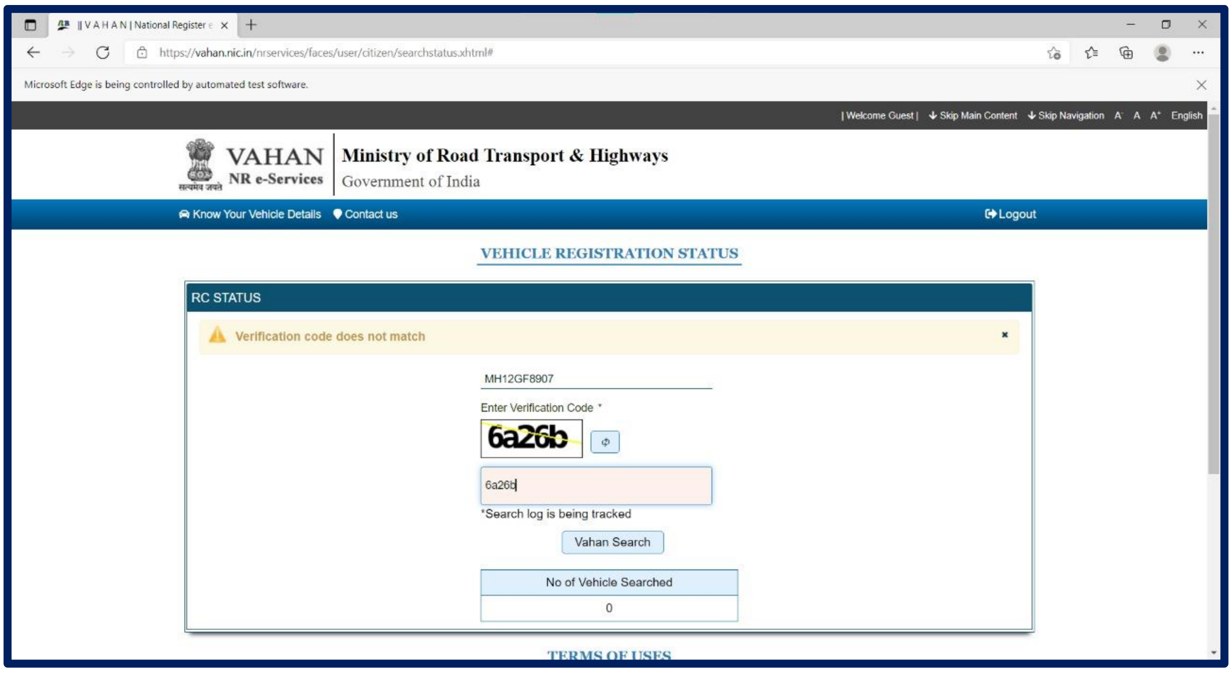


Figure 6: Removal of useless contours in OpenCV.



|| V A H A N | National Register e
https://vahan.nic.in/nrservices/faces/user/citizen/searchstatus.xhtml#

Microsoft Edge is being controlled by automated test software.

| Welcome Guest | Skip Main Content Skip Navigation A- A A+ English
VAHAN
NR e-Services
Ministry of Road Transport & Highways
Government of India
Know Your Vehicle Details
Contact us
Logout

VEHICLE REGISTRATION STATUS

RC STATUS

Verification code does not match

MH12GF8907

Enter Verification Code *

6a26b

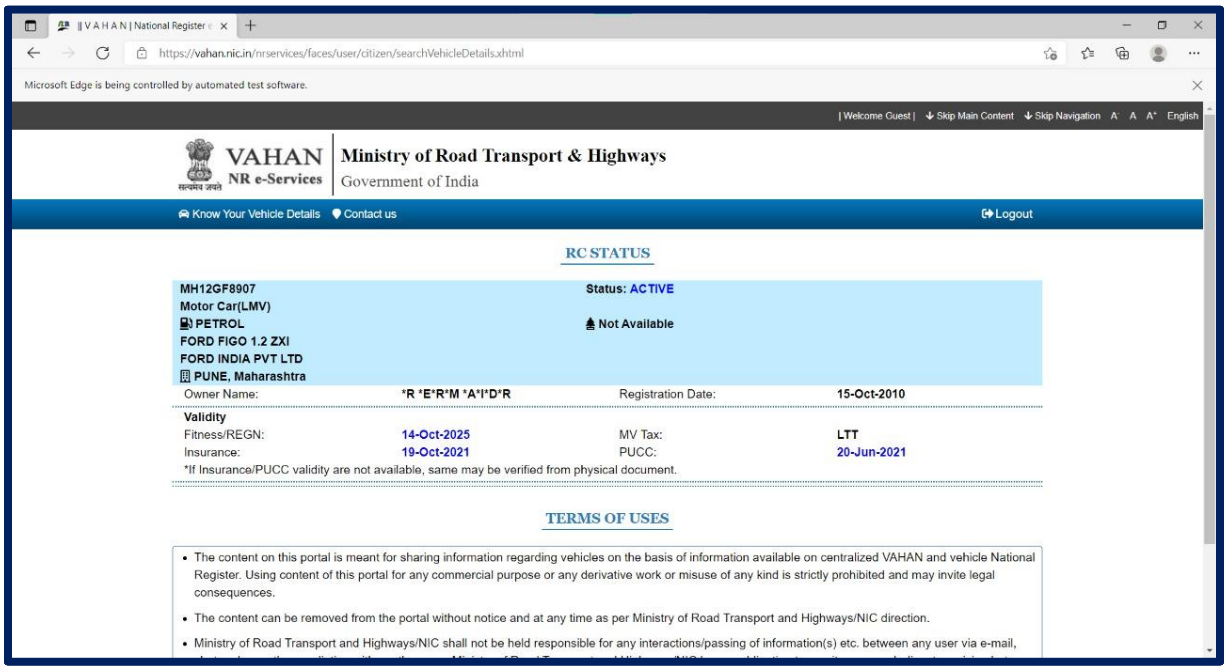
6a26b

*Search log is being tracked

No of Vehicle Searched
0

TERMS OF USES

Figure 7: Using Selenium controlled Edge web driver to scrape details and OpenCV & Tesseract OCR for solving the captcha.



|| V A H A N | National Register e
https://vahan.nic.in/nrservices/faces/user/citizen/searchVehicleDetails.xhtml

Microsoft Edge is being controlled by automated test software.

| Welcome Guest | Skip Main Content Skip Navigation A- A A+ English
VAHAN
NR e-Services
Ministry of Road Transport & Highways
Government of India
Know Your Vehicle Details
Contact us
Logout

RC STATUS

MH12GF8907 Status: ACTIVE
Motor Car(LMV)
PETROL Not Available
FORD FIGO 1.2 ZXI
FORD INDIA PVT LTD
PUNE, Maharashtra

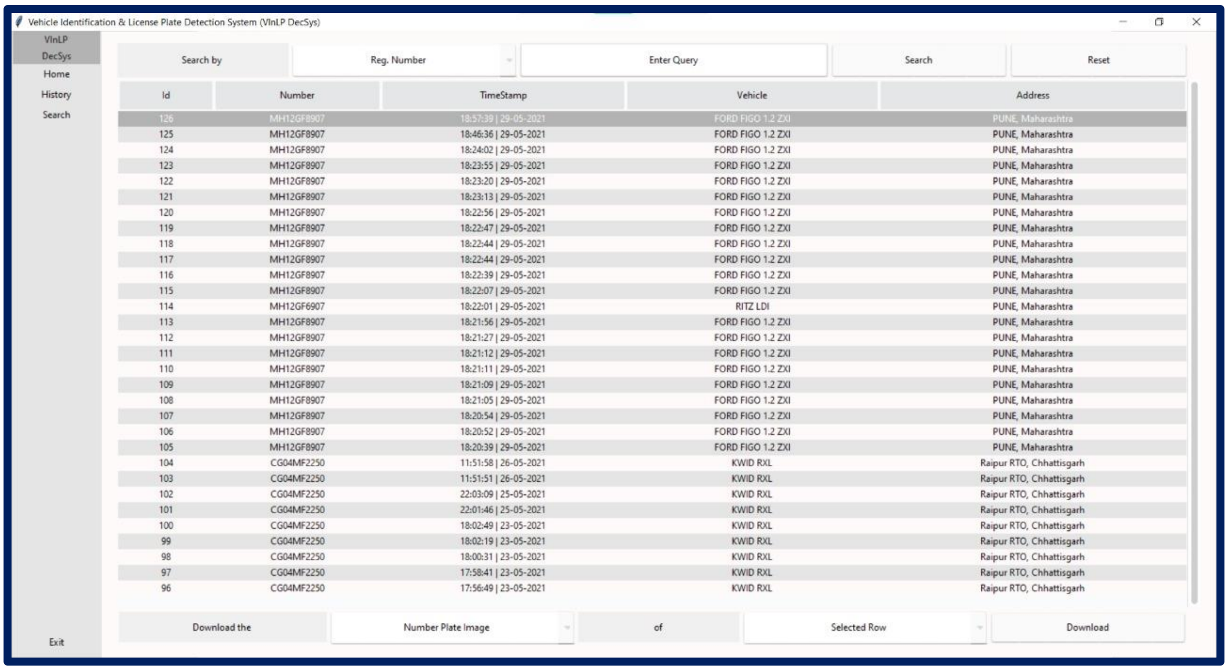
Owner Name:	*R *E*R*M *A*I*D*R	Registration Date:	15-Oct-2010
Validity			
Fitness/REGN:	14-Oct-2025	MV Tax:	LTT
Insurance:	19-Oct-2021	PUCC:	20-Jun-2021

*If Insurance/PUCC validity are not available, same may be verified from physical document.

TERMS OF USES

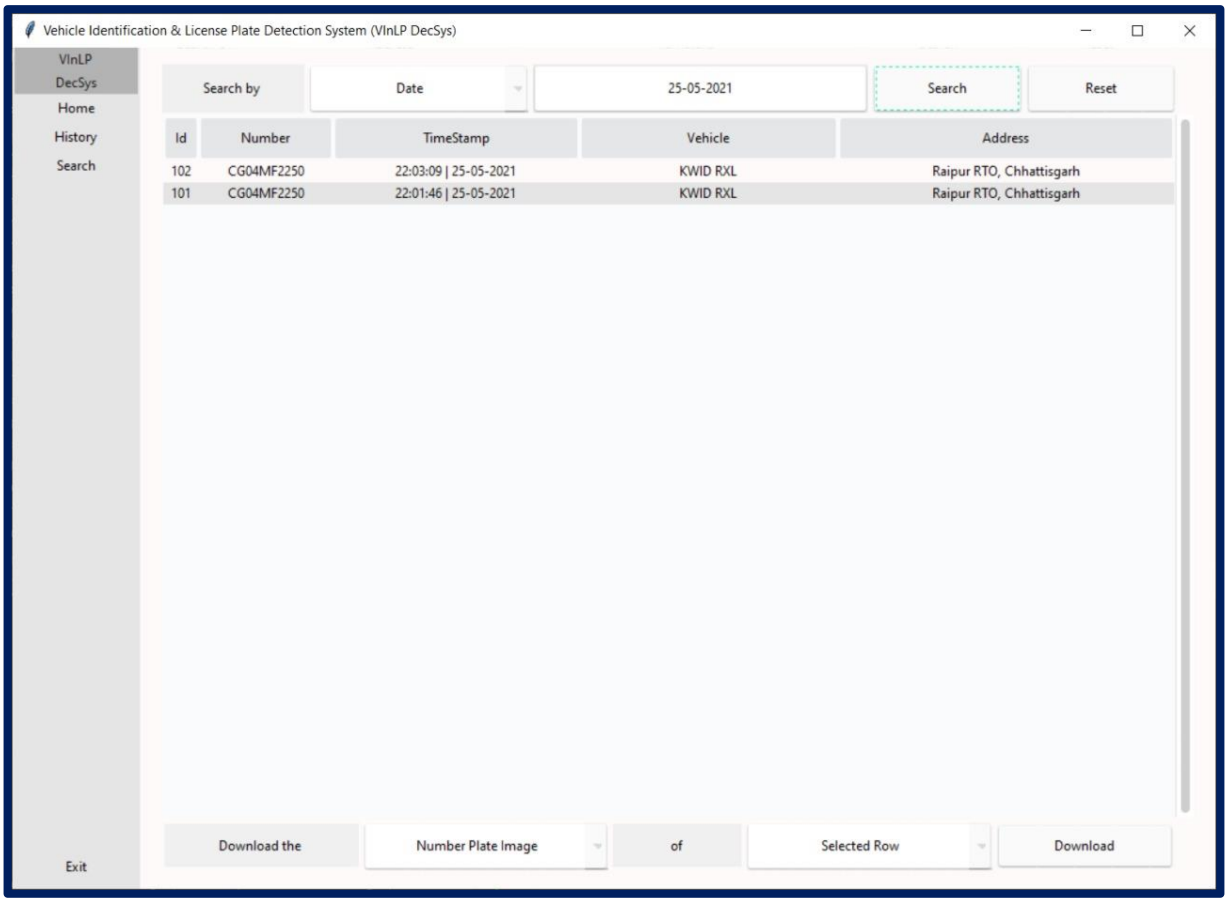
- The content on this portal is meant for sharing information regarding vehicles on the basis of information available on centralized VAHAN and vehicle National Register. Using content of this portal for any commercial purpose or any derivative work or misuse of any kind is strictly prohibited and may invite legal consequences.
- The content can be removed from the portal without notice and at any time as per Ministry of Road Transport and Highways/NIC direction.
- Ministry of Road Transport and Highways/NIC shall not be held responsible for any interactions/passing of information(s) etc. between any user via e-mail,

Figure 8: Selenium - Vehicle details to be scraped.



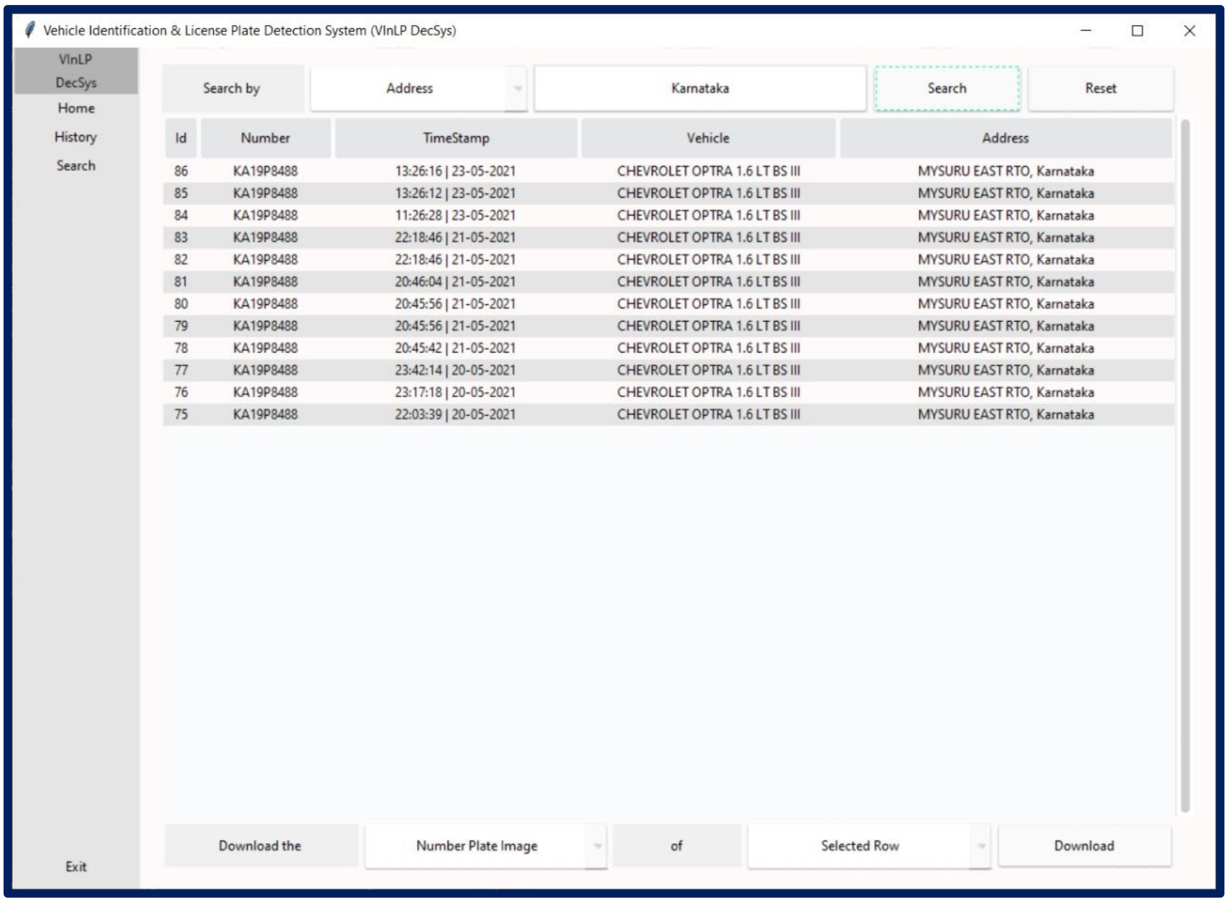
Id	Number	TimeStamp	Vehicle	Address
126	MH12GF8907	18:57:39 | 29-05-2021	FORD FIGO 1.2 ZXI	PUNE, Maharashtra
125	MH12GF8907	18:46:36 | 29-05-2021	FORD FIGO 1.2 ZXI	PUNE, Maharashtra
124	MH12GF8907	18:24:02 | 29-05-2021	FORD FIGO 1.2 ZXI	PUNE, Maharashtra
123	MH12GF8907	18:23:55 | 29-05-2021	FORD FIGO 1.2 ZXI	PUNE, Maharashtra
122	MH12GF8907	18:23:20 | 29-05-2021	FORD FIGO 1.2 ZXI	PUNE, Maharashtra
121	MH12GF8907	18:23:13 | 29-05-2021	FORD FIGO 1.2 ZXI	PUNE, Maharashtra
120	MH12GF8907	18:22:56 | 29-05-2021	FORD FIGO 1.2 ZXI	PUNE, Maharashtra
119	MH12GF8907	18:22:47 | 29-05-2021	FORD FIGO 1.2 ZXI	PUNE, Maharashtra
118	MH12GF8907	18:22:44 | 29-05-2021	FORD FIGO 1.2 ZXI	PUNE, Maharashtra
117	MH12GF8907	18:22:44 | 29-05-2021	FORD FIGO 1.2 ZXI	PUNE, Maharashtra
116	MH12GF8907	18:22:39 | 29-05-2021	FORD FIGO 1.2 ZXI	PUNE, Maharashtra
115	MH12GF8907	18:22:07 | 29-05-2021	FORD FIGO 1.2 ZXI	PUNE, Maharashtra
114	MH12GF6907	18:22:01 | 29-05-2021	RITZ LDI	PUNE, Maharashtra
113	MH12GF8907	18:21:56 | 29-05-2021	FORD FIGO 1.2 ZXI	PUNE, Maharashtra
112	MH12GF8907	18:21:27 | 29-05-2021	FORD FIGO 1.2 ZXI	PUNE, Maharashtra
111	MH12GF8907	18:21:12 | 29-05-2021	FORD FIGO 1.2 ZXI	PUNE, Maharashtra
110	MH12GF8907	18:21:11 | 29-05-2021	FORD FIGO 1.2 ZXI	PUNE, Maharashtra
109	MH12GF8907	18:21:09 | 29-05-2021	FORD FIGO 1.2 ZXI	PUNE, Maharashtra
108	MH12GF8907	18:21:05 | 29-05-2021	FORD FIGO 1.2 ZXI	PUNE, Maharashtra
107	MH12GF8907	18:20:54 | 29-05-2021	FORD FIGO 1.2 ZXI	PUNE, Maharashtra
106	MH12GF8907	18:20:52 | 29-05-2021	FORD FIGO 1.2 ZXI	PUNE, Maharashtra
105	MH12GF8907	18:20:39 | 29-05-2021	FORD FIGO 1.2 ZXI	PUNE, Maharashtra
104	CG04MF2250	11:51:58 | 26-05-2021	KWID RXL	Raipur RTO, Chhattisgarh
103	CG04MF2250	11:51:51 | 26-05-2021	KWID RXL	Raipur RTO, Chhattisgarh
102	CG04MF2250	22:03:09 | 25-05-2021	KWID RXL	Raipur RTO, Chhattisgarh
101	CG04MF2250	22:01:46 | 25-05-2021	KWID RXL	Raipur RTO, Chhattisgarh
100	CG04MF2250	18:02:49 | 23-05-2021	KWID RXL	Raipur RTO, Chhattisgarh
99	CG04MF2250	18:02:19 | 23-05-2021	KWID RXL	Raipur RTO, Chhattisgarh
98	CG04MF2250	18:00:31 | 23-05-2021	KWID RXL	Raipur RTO, Chhattisgarh
97	CG04MF2250	17:58:41 | 23-05-2021	KWID RXL	Raipur RTO, Chhattisgarh
96	CG04MF2250	17:56:49 | 23-05-2021	KWID RXL	Raipur RTO, Chhattisgarh

Figure 9: History Page - Database in GUI.



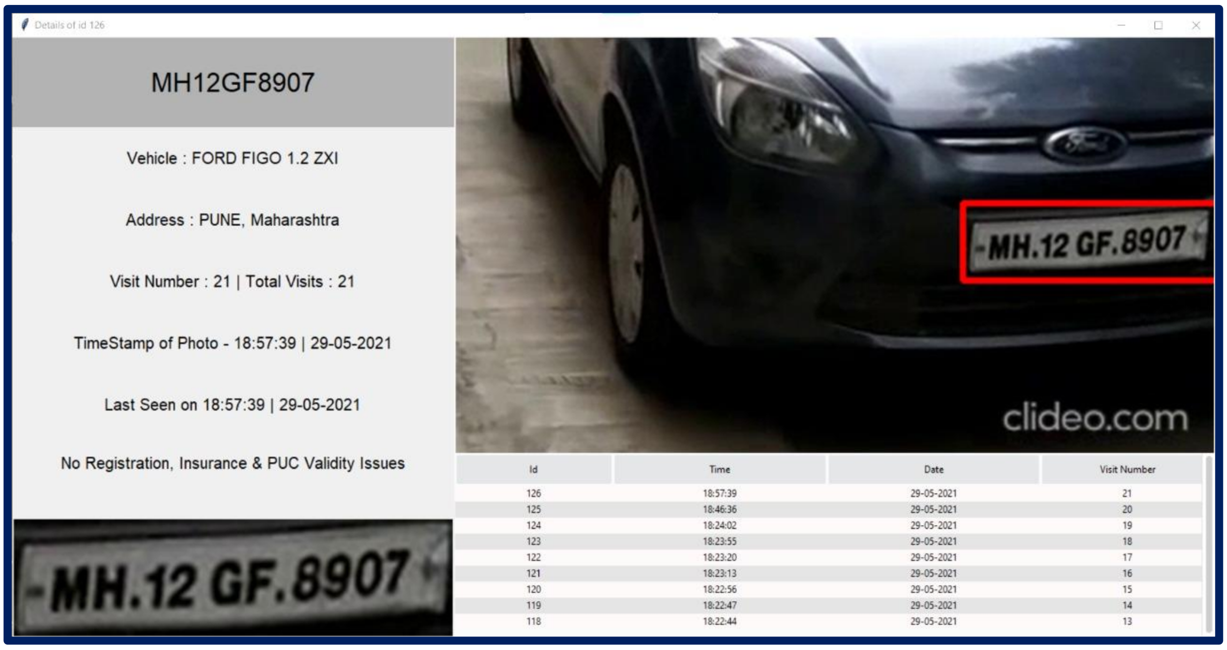
Id	Number	TimeStamp	Vehicle	Address
102	CG04MF2250	22:03:09 | 25-05-2021	KWID RXL	Raipur RTO, Chhattisgarh
101	CG04MF2250	22:01:46 | 25-05-2021	KWID RXL	Raipur RTO, Chhattisgarh

Figure 10: Search feature for particular date.



Id	Number	TimeStamp	Vehicle	Address
86	KA19P8488	13:26:16 | 23-05-2021	CHEVROLET OPTRA 1.6 LT BS III	MYSURU EAST RTO, Karnataka
85	KA19P8488	13:26:12 | 23-05-2021	CHEVROLET OPTRA 1.6 LT BS III	MYSURU EAST RTO, Karnataka
84	KA19P8488	11:26:28 | 23-05-2021	CHEVROLET OPTRA 1.6 LT BS III	MYSURU EAST RTO, Karnataka
83	KA19P8488	22:18:46 | 21-05-2021	CHEVROLET OPTRA 1.6 LT BS III	MYSURU EAST RTO, Karnataka
82	KA19P8488	22:18:46 | 21-05-2021	CHEVROLET OPTRA 1.6 LT BS III	MYSURU EAST RTO, Karnataka
81	KA19P8488	20:46:04 | 21-05-2021	CHEVROLET OPTRA 1.6 LT BS III	MYSURU EAST RTO, Karnataka
80	KA19P8488	20:45:56 | 21-05-2021	CHEVROLET OPTRA 1.6 LT BS III	MYSURU EAST RTO, Karnataka
79	KA19P8488	20:45:56 | 21-05-2021	CHEVROLET OPTRA 1.6 LT BS III	MYSURU EAST RTO, Karnataka
78	KA19P8488	20:45:42 | 21-05-2021	CHEVROLET OPTRA 1.6 LT BS III	MYSURU EAST RTO, Karnataka
77	KA19P8488	23:42:14 | 20-05-2021	CHEVROLET OPTRA 1.6 LT BS III	MYSURU EAST RTO, Karnataka
76	KA19P8488	23:17:18 | 20-05-2021	CHEVROLET OPTRA 1.6 LT BS III	MYSURU EAST RTO, Karnataka
75	KA19P8488	22:03:39 | 20-05-2021	CHEVROLET OPTRA 1.6 LT BS III	MYSURU EAST RTO, Karnataka

Figure 11: Export Excel document /Images feature.



MH12GF8907

Vehicle : FORD FIGO 1.2 ZXI

Address : PUNE, Maharashtra

Visit Number : 21 | Total Visits : 21

TimeStamp of Photo - 18:57:39 | 29-05-2021

Last Seen on 18:57:39 | 29-05-2021

No Registration, Insurance & PUC Validity Issues

Id	Time	Date	Visit Number
126	18:57:39	29-05-2021	21
125	18:46:36	29-05-2021	20
124	18:24:02	29-05-2021	19
123	18:23:55	29-05-2021	18
122	18:23:20	29-05-2021	17
121	18:23:13	29-05-2021	16
120	18:22:56	29-05-2021	15
119	18:22:47	29-05-2021	14
118	18:22:44	29-05-2021	13

Figure 12: Details of particular entry with all the scraped data, images, and processed data like visit number, total visits, validity warnings and previous history of the vehicle.

CONCLUSION

We intend to develop a system which can be used to solve numerous problems related to vehicle regulation, traffic monitoring, surveillance systems etc.

REFERENCES

The following are the references used for building this project.

a. www.analyticsindiamag.com

b. www.opencv.org

c. https://selenium-python.readthedocs.io

d. https://www.github.com/tesseract-ocr/tessdoc/tree/master/README.md

e. https://docs.microsoft.com/en-us/microsoft-edge/webdriver-chromium/?tabs=c-sharp

f. https://docs.python.org/3/library/sqlite3.html

g. https://www.sqlite.org/docs.html

h. https://docs.opencv.org/master/d6/d00/tutorial_py_root.html

i. https://docs.python.org/3/library/tk.html