ASHESI

ASHESI UNIVERSITY COLLEGE

AUTOMATIC LICENSE PLATE RECOGNITION FOR NON COMMERCIAL VEHICLES TO IMPROVE ROAD SAFETY IN GHANA

APPLIED PROJECT

B.Sc. Management Information Systems

Kenneth Mintah Mensah

2016

ASHESI UNIVERSITY COLLEGE

Automatic License Plate Recognition for Non Commercial Vehicles in Ghana to Improve Road Safety

APPLIED PROJECT

Applied Project submitted to the Department of Computer Science, Ashesi University College in partial fulfilment of the requirements for the award of Bachelor of Science degree in Management Information Systems

Kenneth Mintah Mensah

April 2016

DECLARATION

I hereby declare that this Applied Project is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:

………………………………………………………………………………………

Candidate's Name:

………………………………………………………………………………………

Date: ………………………………………………………………………………………

I hereby declare that preparation and presentation of this Applied Project were supervised in accordance with the guidelines on supervision of Applied Project laid down by Ashesi University College.

Supervisor's Signature:

………………………………………………………………………………………

Supervisor's Name:

………………………………………………………………………………………

Date: ………………………………………………………………………………………

Acknowledgement

To my supervisor, Dr. Nathan Amanquah and the computer science department, I am thankful for the mentoring and scholarly advice that you gave to me. I am also thankful to Mr. Nicholas Korblah Tali for his invaluable contribution to this project.

Abstract

Vehicles are ingenuous contraptions that have significantly revolutionized how human beings move from one place to another. However, inasmuch as they have liberated man, they have also become a major cause of fatalities and injuries. Technologies such as Automatic License Plate Recognition (ALPR) have been utilized in advanced countries to make roads safer to use. However, they are not utilized in lesser developed countries such as Ghana to improve road safety. In this project, an ALPR system is implemented for the Ghanaian context. It is intended to recognize and detect number plates of non-commercial vehicles that jump the red light by using a mobile phone camera and Optical Character Recognition (OCR).

Table of Contents

Chapter 1: Introduction

1.1 Introduction

Vehicles are ingenuous contraptions that have significantly revolutionized how human beings move from one place to another. They are without doubt the primary source of transportation and have become so ingrained in our daily lives that they not only aid our inevitable desire to be mobile, but also serve as status icons. However, inasmuch as vehicles have liberated man, they have also become a major cause of fatalities and injuries in the 21st Century. According to the World Health Organization (WHO), it is estimated that road accidents account for 1.25 million deaths every year (World Health Organization, 2015). Again, it is the number one cause of death among people between the ages of 15 and 29. This implies that a significant portion of the world's most productive demographic is lost to road accidents every year.

Alarmingly, while only 54% of registered vehicles in the world are in low and middle income countries, 90% of road fatalities occur in these countries with African countries having the highest percentage of deaths per road accident (World Health Organization, 2015). Multiple reasons can be attributed to such high fatality rates in low and middle income countries such as Ghana. A few of them are the utter disregard for traffic regulations such as jumping the red light, excessive speeding, the use of vehicles that are not roadworthy and the lack of an incorruptible police system to ensure that punitive measure are meted out to offenders.

So far measures to improve road safety in Ghana have proven to be somewhat ineffective as indicated by statistics. Nonetheless, as evidenced in advanced countries, integration of technological innovations and systems in road safety can significantly curb indiscipline on roads and mitigate the effects of road accidents. For instance, in the UK and Spain, Automatic License Plate Recognition (ALPR) is used by the police to detect license plates associated with unlicensed drivers, identify uninsured vehicles, detect stolen license plates and vehicles (Gaumont & Babineau, 2008).

ALPR is a technology that can be used to automatically detect and recognize the license plate of a vehicle without any human intervention. ALPR systems have many applications in the road safety and traffic control. One application is the detection and recognition of vehicles of drivers that jump the red light. However, such systems are absent in many low and middle income countries such as Ghana because of the limited financial resources. Moreover, because of the complexities associate with developing ALPR systems, they have to be developed in context for the specific country or region it is intended for.

This project attempts to reconcile the work and assumptions of developers in advanced countries to develop an ALPR system for the Ghanaian driving context that will capture and identify the number plates of vehicles that jump the red light. Upon successful detection and recognition, the system will immediately send a record of the offender’s number plate to traffic law enforcers and road safety administrators so that they can be appropriately fined or arrested in Ghana. The system will focus primarily on non-commercial vehicles, which have the white number plate. Also as an initial version, it will be limited to non-slanted images.

1.2 Motivation

In developing countries, the death toll and casualties caused by road accidents is devastating. In Ghana, road accidents are the second major cause of death. In 2009, on average, 6 people died by road accidents (Oppong, 2012). Among the myriad causes of road accidents, accidents caused by drivers who jump the red light have been shown to be quite catastrophic. Due to limited resources and technology, traffic law enforcers in Ghana find it difficult to keep track of traffic light offenders and mete out punitive measures to deter other offenders. Also, traffic light offenders easily evade the law when there are no traffic law enforcers stationed at traffic lights. Moreover, traffic law enforcers have garnered the reputation for being corrupt; hence offenders who are willing to bribe the enforcers are easily let off and no record of their crime is kept (Oppong, 2012). Having identified this problem, this project seeks to develop a solution that could ultimately make Ghanaian roads safer to use.

1.3 Problem Statement

Notwithstanding the numerous attempts made to reduce the death toll caused by road accidents in Ghana, road users still run a high risk of losing their lives or suffering life-threatening injuries everyday. Unfortunately, traffic law enforcers are not incorruptible hence motorists who break traffic regulations are not dealt with appropriately. Technologies, such as ALPR can be used to make roads safer. However, they are expensive and have not been designed for Ghanaian number plates due to the complexities associated with developing them.

1.4 Benefit of Proposed System

An ALPR system for the Ghanaian roads would significantly improve road safety in that motorists who do not abide by the traffic regulations can be instantly notified and fined without any human intervention. Moreover, appropriate sanctions will be meted out to violators of traffic regulations since they will be no human intervention between when the violators are recognized and when the fines are paid. Additionally, the data gathered from the system could be used to identify frequent violators and they can be blacklisted and kept off the streets. An added benefit would be that such information, if available to insurance companies can influence insurance premiums and serve a disincentive for bad driving behaviour on the roads. Ultimately, the system would be an unbiased law enforcer designed with the sole aim of making Ghanaian roads safer.

1.5 Project Objectives

The focus of this project is to build a cost effective ALPR system to aid traffic law enforcers keep drivers in check and ultimately reduce the occurrence of road accidents caused by vehicles jumping the red light Ghana. Project objectives are:

- Design and implement a system to detect and recognize number plates
- Design and implement a system to notify drivers of their offence and the associated fine
- Implement an application that could be used to transfer captured images to an image processing server

1.6 Overview of Chapter

This paper comprises six chapters. In this chapter, readers are introduced to the project and the rationale or motivation for this project. Chapter 2 discusses prior research that has been conducted with regards to ALPR as well as key concepts required to aid readers gain better understanding of ALPR systems. It also highlights existing solutions and discusses the technologies available to develop the proposed system. Chapter 3 presents to the reader a high level description of the system and the requirements that the system must meet. In chapter 4, the methodology that will govern the design and implementation of the system is discussed. In addition, the architecture governing the design of the system is detailed in this chapter. Chapter 5 describes the implementation process that was used to develop the system and the tests that were conducted on the system. In chapter 6, the results of the proposed system are discussed and evaluated. Finally, in chapter 7, the limitations of the system are presented and further work required to improve the system are suggested.

Chapter 2: Background, Related Work & Technologies

This chapter gives an introduction to the concept of Automatic License Plate Recognition in Section 2.1. The subsequent section, Section 2.2, describes key concepts and techniques used in developing ALPR systems. Then, it discusses existing solutions in Section 2.3. In Section 2.4, a high level description of the proposed system is given to the reader. Finally, platforms, frameworks and services that could be used to develop the proposed system are discussed in in the subsequent sections of this chapter.

2.1 Background

In order to gain an appropriate understanding of ALPR systems, it is necessary to discuss existing research and projects that have been conducted in this area.

Generally, ALPR systems comprise four processing stages or modules. These are the image acquisition stage, the license plate extraction stage, the license plate segmentation stage and the character recognition stage (Solanki, Rai, & Teena, 2013) as shown in Figure 2.1. In the image acquisition stage, key things to consider are camera resolution and shutter speed since most Optical Character Recognition(OCR) software recommend a minimum image resolution of 300 dots per inch(dpi) for effective data extraction and meaningful output (Neal & Kevin, 2010). The license plate extraction stage involves identifying the location of the number plate in the captured image and extracting it. In the license plate segmentation process, probable license plate characters are segmented. Finally, the segmented characters acquired from the previous stage is recognized.

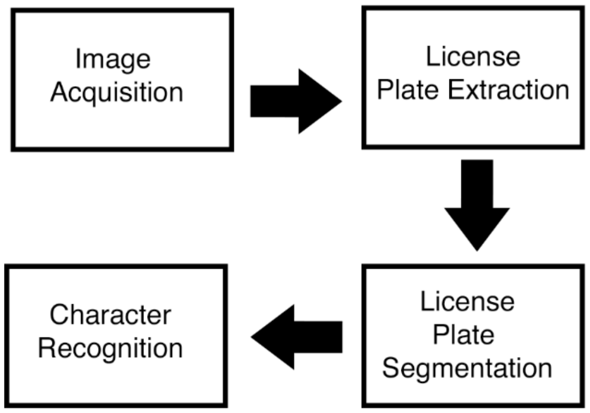


Figure 2.1: The four processing stages of ALPR systems.

Myriad academic projects revolve around developing ALPR systems primarily because there are several challenges involved in developing this system. Some of these challenges are noisy images, slanted images and poor lighting conditions. Consequently, several approaches or methodologies have been proposed for developing ALPR systems, some of which are discussed in subsequent paragraphs.

In Korea, a system was designed specifically for Korean license plates using a Support Vector Machine (SVM) based character recognizer. This system was tested on 1000 video sequences and achieved a 100% car detection rate, a 97.5% segmentation rate and a 97.2% character recognition rate (Kim, Kim, Kim, & Kim, 2000).

Yanamura et al. (2003) proposed another method of license plate tracking and extracting from images taken sequentially over time by a video camera. They employ Hough transform and Vector Block Matching as techniques for extracting and tracking the number plate with a good degree of accuracy. Similarly, Huang, Lai and Chuang (2004) employed a video camera for image acquisition. Subsequently, they utilize a gradient operator to localize or identify probable areas of a number plate. Huang et. al (2004) also use the Otsu thresholding method to binarize the images and subsequently template matching for character recognition.

Quite contrary to the template matching approach for OCR proposed by Huang et. al (2004), Zhai, Bensaali and Sotudeh (2012) presented an Artificial Neural Network(ANN) based OCR algorithm for ALPR systems. This algorithm was tested on a database of 3700 UK binary character images and has been shown to meet the real-time requirement of ALPR systems with an accuracy of 97.3%.

Duan, Duc and Du (2004) employed boundary line based approach by combing Hough transform algorithm and contour algorithm. Their research indicates that this combination of algorithms for vehicle license plate recognition optimizes speed and accuracy compared with morphology-based and texture-based implementations. Duan et al. (2004) evaluated their algorithm on two sets of Vietnamese datasets and obtained an accuracy of approximately 99%.

Kasaei and Kasaei (2011) propose a robust and real-time method for license plate detection and recognition using a combination of template matching and morphological operations. Their system is tested on license plates obtained from the Iranian traffic control organization. Kasaei and Kasaei (2011) obtained a 98.2% accuracy on license plate detection and a 92% accuracy on character recognition.

Yoshimori, Mitsukura, Fukumia and Akamatsu (2003) propose a robust method for dealing with license plate detection for vehicles moving at very high speeds and on a rainy day. They use the real coded genetic algorithm (RGA) to determine the most likely number plate colours under various lighting conditions.

As discussed, there are many approaches to developing ALPR systems. In praxis, however, the project cannot employ all of these techniques and approaches. For this reason, some of the

techniques that will be employed include Otsu thresholding as used by Huang et. al (2004) and morphological operations employed by
Kasaei and Kasaei (2011).

2.2 Key Concepts and Techniques Used

In the preceding section existing research and approaches were discussed. In this section, key concepts and techniques used in developing ALPR systems and image processing are explained.

2.2.1 Morphological Operations

Morphological operations are image processing techniques or algorithms that deal with shape of images in an image (Goyal, 2011). They are usually applied to images to improve quality and remove imperfections after segmentation. There are four basic morphological operations: erosion, dilation, opening and closing.

Dilation is used to repair breaks and intrusions in an image by thickening binary objects (Figure 2.2(b)). Erosion is used for shrinking and thinning binary objects (Goyal, 2011) (Figure 2.2(a)). Opening is obtained by eroding an image and then dilating it. It is used to remove small white regions in an image. Closing is the inverse of opening. First the image is dilated then it is eroded. It produces the opposite effect of opening (Brahmbhatt, 2013, p. 63).

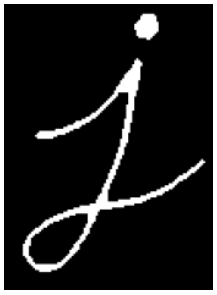


Figure 2.2(a): Image showing erosion
Source: (Brahmbhatt, 2013)

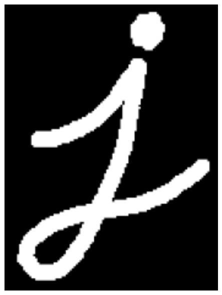


Figure 2.2(b): Image showing dilation

2.2.2 Edge Detection

Edge detection is a critical step in object recognition. It involves finding sharp discontinuities or abrupt changes in pixel intensity in objects in an image. Fundamentally edge detection produces an image outlining objects in an input image. Presently, the two main edge detection algorithms are Sobel Edge detection and Canny Edge detection. With Sobel Edge, a convolution matrix is applied to the input image which produces an output image showing the identified edges in the image. Similarly, with Canny Edge a convolution matrix is applied to the image. However, a Gaussian blur operation is applied to the input image before the convolution occurs. This results in an output image with less noise compared to the one produced with Sobel Edge detection (Mordvintsev & Abid, 2013).

2.2.3 Template Matching

Template matching is a technique used in classifying an object or recognizing characters by comparing a portion of an image with a reference image. Essentially, features in an image are compared with those in a reference image to identify similarities. There are two main approaches used in template matching: the feature-based approach and the area-based approach. The feature-based approach is more suitable when the template and reference images have more features in common. The area-based approach is a combination of feature detection and feature matching. It is best suited for images that have no strong features with the reference image (Mahalakshmi, Muthaiah, & Swaminathan, 2012).

2.2.4 Artificial Neural Networks (ANN)

Artificial Neural Network (ANN) is an intelligent system that can be used to recognize a character based on its topological features such as symmetry, shape or number of pixels. ANN is a robust system that can be used to train sample characters to recognize characters of interest (Shrivastava & Sharma, 2012). ANN provides a robust method for analyzing the line of discontinuity between characters and identify stroke edges in an image. The main advantage ANN has over template matching with regards to character recognition is that ANN systems can always be trained to recognize specific characters. ANN is an approach used in developing and improving the accuracy of OCR systems.

2.2.5 Binarization

Binarization is the process of converting pixels in an image into only black and white using a threshold of values. Binarizing images involves utilizing thresholding algorithms. There are mainly two types of thresholding algorithms: the global thresholding algorithms, which classify and convert image pixels to white or black base of a user specified range, or cutoff and the adaptive thresholding algorithms which converts pixels to white or black based on the values of neighboring pixels. In developing robust systems, adaptive thresholding algorithms are used. One of such algorithms is the Otsu binarization which comparatively produces a much better binarized image (Sauvola & PietikaKinen, 2000).

2.2.6 Optical Character Recognition (OCR)

Optical Character Recognition (OCR) is the process of recognizing or identifying characters in an image. In that regard, OCR engines are systems that can be used to identify characters in an image by a computer. OCR systems utilize ANN, template matching or recognition using correlation coefficients (Shrivastava & Sharma, 2012).

2.3 Existing Solutions

In this section, some existing ALPR systems are discussed.

2.3.1 Java ANPR

Java ANPR is an open source number plate recognition library developed with Java. The system explores techniques in artificial intelligence, machine vision and neural networks to develop the automatic number plate recognition system (Martinsky, 2007). It uses a collection of still images captured with a camera. Java ANPR has the following recognition rates:

Table 2.1: This table show the recognition rates of the Java ANPR system

	Total number of plates	Total number of characters	Weighted Score (%)
Clear plates	68	470	87.2
Blurred plates	52	352	46.87
Skewed plates	40	279	51.64
Average plates	177	1254	73.02

Source: (Martinsky, 2007)

2.3.2 OpenALPR

OpenALPR is an open source Automatic License Plate Recognition library originally developed in C++. It has wrappers and bindings in C#, Java, Node.js, Go and Python. The library analyzes both video streams and still images and outputs a text representation of any identified number plates (OpenALPR, 2016). OpenALPR can be deployed on Windows, Ubuntu Linux, OS X, Android and iOS. The library also has a cloud API for developers. It provides support for license plates in North America, Europe, Australia, Great Britain, Singapore and Korea. It does not provide support for Ghana's license plates. OpenALR also comes with a commercial license for business and institutions priced at $50 per camera per month (OpenALPR, 2016).

2.4 Platforms

To develop an ALPR system a number of platforms or services will be required. This section describes some of these platforms and services that can be used to develop an ALPR system.

2.4.1 Android

Android is currently the most pervasive OS in the world. It is an open source Linux-based platform for mobile devices released under the Apache 2 license. It was developed by Google and the Open Handset Alliance (OHA), which comprises more than 30 hardware, software and telecommunications companies. Native android applications can be developed with the Dalvik programming language which is a variation of Java or a combination of Java and Xml (Rouse, 2008). In this project, the minimum software development kit (sdk) version that will be used is 14.

2.4.2 Python

Python is a very powerful and dynamic object oriented programming language. As an interpreted language, it is comparatively fast. Compared to other object oriented programming languages such as Java and C++, python is easy to learn. Moreover, developers spend less time over syntactic details and more time on the functionality of algorithm. This makes it a fast development tool especially for developing large systems. Additionally, python can be easily integrated with other programming languages making it very easy to work with. The version used in this project was 2.7.1.

2.5 Preprocessing Libraries

2.5.1 OpenCV

OpenCV is an open source image processing library originally developed in C and C++. However, bindings in Java and Python are available for developers who are a more proficient in these languages. It offers real-time processing of images and extensive support for developers. OpenCV has a learning curve and requires fundamental understanding of image processing techniques (Rosebrock, 2014).

2.5.2 Scikit-Image

Scikit-Image is a collection of image processing algorithms developed in python. It is an open source tool developed by the SciPy community (SciPy, 2016). It offers functionality for image processing techniques such as histogram equalization, grayscale conversion image filtering and a host of other image processing operations.

2.5.3 NumPy

NumPy is an extensive library for scientific and mathematical computing with the Python programming language. The library is licensed under the BSD license; hence it can be reused with few restrictions (NumPy, 2016). It provides extensive support for large, multidimensional arrays. With NumPy, images can be expressed as multidimensional arrays. This makes it computational and resource efficient to work with images (Rosebrock, 2014).

2.6 OCR Libraries

2.6.1 Tesseract OCR

Tesseract OCR is an open source document analysis and OCR system used to recognizing text in binarized images. Originally developed at HP between 1984 and 1994, Tesseract OCR became a more recognized OCR engine after it emerged one of the best OCR libraries in the 1995 UNLV Annual Test of OCR Accuracy. As a widely recognized open source OCR library, Tesseract has become quite powerful. It is currently maintained by Google. Tesseract was originally developed in C++ but has wrappers and bindings for Java, Python, PHP, Ruby, .Net, Objective-C and Clojure. It can be deployed on Windows, Linux and Mac OSX operating systems. Tesseract is a multilingual script recognition system with support for over 30 languages. It supports various output formats such as plain-text, html and pdf (Tesseract OCR, 2016). Tesseract OCR requires preprocessing in order to attain the desired accuracy. Additionally, lower resolution images with a resolution lower than 300 ppi produce quite low recognition rates. Nonetheless, Tesseract is still widely used because of its high accuracy rates when the preconditions have been met.

2.6.2 Java OCR

Java OCR is a suite of pure Java libraries for document analysis and character recognition. Like Tesseract OCR, it is an open source library and can be trained. It can be deployed on any OS because it is developed in Java. Compared with other extensively used OCR libraries, Java OCR has a lower accuracy rate.

2.6.3 OCRopus

OCRopus is a collection of free document analysis programs and libraries released under the Apache 2 license (Ocropy, 2016). OCRopus is developed for Linux. However, it can also be deployed in unix-based operation systems such as Mac OS X. OCRopus also offers multilingual script recognition but for a lesser collection of languages compared with Tesseract OCR. Moreover, testing reveals lower accuracy compared with Tesseract OCR.

2.7 SMS Services

2.7.1 SMS GH

SMSGH is an SMS gateway that makes it possible to send and receive text messages from a network operator in Ghana. It can be integrated into applications or systems to provide SMS features for those systems. SMSGH requires internet connection and an account that will be used to pay for SMS transactions.

2.7.2 FrontlineSMS

FrontlineSMS is a desktop and mobile application that is used to enable instantaneous two-way communication to any mobile device. It is offers good support for receiving bulk SMS messages (FrontlineSMS, 2016). It is currently the world's most popular text messaging software. Unlike SMSGH, it is free, open source and does not require the Internet. However, the airtime of the connected mobile will be used if sending messages.

2.8 Proposed Solution

This project identifies the lack of ALPR systems for recognizing Ghanaian number plates and develops a cost effective ALPR system for the Ghanaian number plates. It utilizes already established algorithms and techniques to develop one suited to identifying characters in the Ghanaian number plate. It will include an SMS notification system to notify traffic law violators that their offence has been noticed and inform them of the fine corresponding to their offence.

2.9 Scope

The system to be developed, here after referred to as the ShutterCop ALPR system is intended to be used to aid the traffic law enforcers to automatically detect and recognize license plates of vehicles that jump the red light. It is presently limited to identifying only non-commercial license plates (white background). It cannot recognize characters in a skewed image. In subsequent chapters the proposed system is described in further detail.

2.10 Summary of platforms and components

In previous sections, platforms, libraries and services such as SMS gateways have been discussed with the intention of exploring available technologies for developing the proposed system. The Android platform in combination with Python is the primary platform used in implementing the ShutterCop ALPR system. Complementary image processing libraries such as OpenCV and Numpy utilized heavily both during preprocessing and actual processing. SMSGH Application Program Interface(API) will be utilized as a notification tool instead of Frontline SMS since the former SMS service is comparably faster for sending text messages and bulk SMS.

2.11 Summary of Background, Techniques and Technologies

Prior to this chapter, ALPR systems were introduced in a larger context. In this chapter, however, ALPR systems have been discussed in a more technical context. Techniques, platforms, libraries and technologies have been expounded in this chapter to afford the reader a deeper understanding of ALPR systems and what can be used to implement such systems. In the next chapter, the requirements specifications of the proposed system are discussed.

Chapter 3: Requirements Specification

The purpose of this chapter is to provide a detailed description of the functionalities of the ShutterCop ALPR System. This chapter describes the project's target users, hardware and software requirements.

3.1 Project Scope

ShutterCop ALPR System comprises two main components: a server-side application to process captured images of number plates and a client-side application which will either run on Android handsets or web-based. The system is designed to facilitate the activities of traffic law enforcers in Ghana by enabling them to instantly detect and fine drivers that jump the red light without having to physically pursue them.

3.2 Overall Description

3.2.1 Product Perspective

The ShutterCop project is a cost-effective application of the Automatic Number Plate Recognition (ANPR) system used in advanced countries. It is primarily intended to run on a server. However, there will be two client side applications which will either be web-based or developed on the android platform. The server-side component will be responsible for image processing and database services. Additionally, the client-side application will be a platform for traffic light offenders to pay their fines. Below is a diagram of the ShutterCop system which illustrates the how the client and server applications interact.

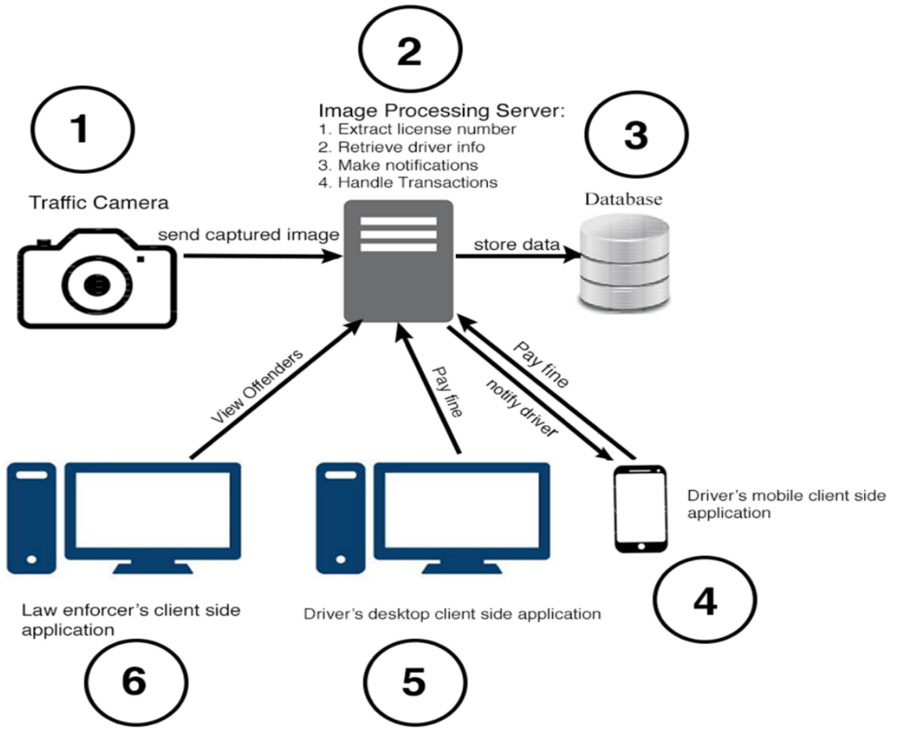


Figure 3.1: High Level Overview of subsystems and Components of the ShutterCop ALPR systems.

3.2.2 Product Features

The following list offers a brief outline and description of the features and functionalities of the ShutterCop system. The features have been subdivided into core features and additional features. Core features are the necessary features of the application whereas additional features provide extra features. Therefore, the latter features will only be implemented when the core features are fully functional.

3.2.2.1 Core features

1. Image capturing & transmission to server:
    - Captures images of vehicles that cross the red light

- Transfers the image to the ShutterCop server

2. Number plate recognition and OCR:
    - Receives captured images of number plates
    - Filter images
    - Detect edges of number plates in vehicles
    - Recognizes or identifies the vehicle number

3. Offender information retrieval:
    - Match vehicle owner to number plate
    - Records vehicle owner offence
    - Records the fine associated with offence and vehicle owner

4. SMS notification of offender:
    - Automatically delivers notifications to vehicle owner

3.2.2.1 Additional features

5. PayPal/mobile money integration:
    - Includes a system for offenders to pay their fine online
    - Enables secure and fast transactions between offenders and the Motor Transport and Traffic Unit (MTTU)
    - Automatically updates offender's history and outstanding balance of fines

6. Online receipt system:
    - Issue online receipt to offenders who pay their fines online

7. Offender history visualization:
    - Shows the driving history of an offender
    - Shows the fines that an offender has not paid or paid
8. RESTFUL API for insurance companies:
    - Receive details about the driving history of a driver.

3.2.3 User Classes and Characteristics

The ShutterCop ALPR Project is meant to aid traffic law enforcers and the MTTU keep track of drivers who jump the red light without having to engage in a hot pursuit with them. It will do so by capturing images of vehicles that jump the light and instantly sending drivers a notification that they have been fined. The application will primarily exist on the server. Hence, it will have no learning curve. The user interface for viewing offenders and paying fines will be as intuitive as possible and very simple. The main user types are listed below:

1. Traffic light enforcers
2. Drivers

From the list of users above, only traffic law enforcers should have access to information about traffic light offenders. Hence, traffic law enforcers will have special privileges to view but not edit information on drivers. Drivers will only have access to a payment portal so that they can pay their fine or review their fine history.

3.2.4 Operating Environment

The main component of the ShutterCop ALPR System is the server side application which will be used for processing captured images of vehicle numbers. The system is graphics intensive so the capability of the hardware must be taken into consideration. The system will rely on several functionalities that OpenCV offers. This requires at least basic understanding of software development in C++, Python or Java. The server side application will be developed in python.

3.2.5 Design and Implementation Constraints

There are quite a number of constraints associated with implementing this system. The quality of the image captured can significantly affect the accuracy of the OCR engine and algorithm. Also, slanted images have an effect on the ability of the OCR engine to detect numbers. Also only one type of number plate can be used to develop the system within the required time frame. Moreover, the system requires substantial processing power, memory and storage for processing and storage of captured images. ShutterCop must be quick and responsive, even when processing multiple images and handling transactions, so the system must be designed efficiently. The system will be developed in python to reduce verbosity and make the system easy to debug. This could however increase the overhead of the program.

3.2.6 Assumptions and Dependencies

3.2.6.1 Time dependencies

Due to time constraints development of core features will be prioritized over extra features. Additional features will be developed only when the core features are fully functional. The system is designed only for vehicles with a white background number plate since they are the only plates currently available for testing.

3.2.6.2 Hardware dependencies

One critical component of this system will be the camera that will be used to capture the images. The resolution of the camera will determine the algorithm and hence the overhead and efficiency of the algorithm. Image processing requires a lot of processing power hence the system will depend on the processing power and memory of the server in order to be efficient and quick.

3.2.6.3 External dependencies

- Email notifications

The server will be responsible for sending email notifications to the drivers and traffic law enforcers. Once the number plate has been successfully extracted the server will send the driver an email notification. This feature will be developed with either PHP or Python.

- SMS notifications

Offline drivers without an Android smartphone will be notified of fines and infractions via text messages. A free text messaging API is yet to be found.

- Tesseract OCR

Tesseract OCR engine will be used together with OpenCV to extract the number plate from the images using OCR.

- OpenCV

For edge detection of number plates and preprocessing of images to enable extraction of numbers, OpenCV library will be used.

3.3 System Features

ShutterCop ALPR's system features comprise two main categories: core and additional features. Core feature are the rudimentary part of the system and are essential to the system. These features will be prioritized over additional features until they are fully functional. Additional features are totally dependent on the time constraint and will be added as time permits.

3.3.1 Core Features

3.3.1.1 Image capturing & transmission to server

The camera will be triggered to capture images of vehicles when they cross the red light. This image is then sent to the server-side application for processing and storage.

Stimulus/response sequences

Step 1 : Camera is triggered to capture image of vehicles

Step 2 : The image is sent to the server and stored in the database

System requirements

- Secure database system: The application must ensure that captured images are securely stored and cannot be hacked by computer savvy offenders who want to evade the law.

- High resolution cameras: The system should capture high resolution images that the server can efficiently process.

- High speed internet: The system should have good and fast internet connectivity to the server to transmit the images to server quickly for processing.

3.3.1.2 Number plate recognition and OCR

This will be the main feature of the application extracting the number plate(s) from the captured image with a high degree of accuracy.

Stimulus/response sequences

Step 1 : Captured Image is filtered to remove noise

Step 2 : An algorithm is used to detect the edge of the number plate

Step 3 : OCR engine kick in to extract the number from the plate

System requirements

- Image filtering algorithm: The algorithm must filter the image and clear noise that can affect the accuracy of the OCR engine.

- Edge detection algorithm: The algorithm must detect the edges of the number plate in the image so that the OCR Engine can work only within that area.

- OCR algorithm: The algorithm must with a high degree of accuracy extract the number from the number plate.

3.3.1.3 Offender information retrieval

This feature is responsible for matching and retrieving information from the drivers' database and matching it to the vehicle number of the offender.

Stimulus/response sequences

Step 1 : Search for vehicle number in the database

Step 2 : Show corresponding owner of vehicle or driver

System requirements

- Search algorithm: This algorithm must efficiently search the database for the corresponding driver of the vehicle.

3.3.1.4 SMS notification of offender

This feature sends the traffic light offender or driver an SMS notification with the specified fine included.

Stimulus/response sequences

Step 1 : When the driver of the associated vehicle is found their phone number is retrieved and an SMS message is composed.

Step 2 : Send driver an email with the enclosed fine amount

User requirements

- A valid phone number: The offline user must have some method of being contacted via SMS

System requirements

- SMS capabilities: The server must be able to compose and send email and SMS messages

3.3.2 Additional Features

3.3.2.1 Offender history visualization

This feature allows traffic law enforcers to view the history of offenders. Traffic law offenders are able to see offenders who have not paid their fines as well as the frequency of their offence.

Stimulus/response sequences

Step 1 : The traffic law enforcer logs into the system.

Step 2 : From here the user can view information about recent offenders and offenders who have not paid their fines.

System requirements

- GUI implementation: The application must have a method of converting raw data into rich dynamic visuals.

3.3.2.2 PayPal/mobile money integration

This feature will offer drivers the option of paying their fine through PayPal or Mobile Money. This will enable users to make real financial transactions through the system and update their driver history accordingly.

Stimulus/response sequences

Step 1 : The driver accesses this feature from any transaction-related page on the mobile or web app.

Step 2 : From here the user securely pays the fine for the offence they committed online.

Step 3 : The user confirms the transaction and signs out.

User requirements

- Paypal/mobile money account: Only users who have a PayPal account or Mobile Money will be able to utilize this functionality.

System requirements

- Paypal/mobile money integration: The application must utilize the PayPal API to provide the appropriate functionalities.

3.3.2.3 Online receipt for traffic offenders

Stimulus/response sequences

Step 1 : The user accesses the transaction-related page.

Step 2 : From here the user can view information about their payments and receipts and print.

System requirements

- GUI implementation: The application must have a method of converting raw data into rich dynamic visuals.

3.3.2.4 RESTFUL API for Insurance Companies

Stimulus/response sequences

Step 1 : The user application sends a request to the API using their secret and key

Step 2 : If the user is authentic, they are given access to the driver details requested

User requirements

- User secret and key: The application must have a method of converting raw data into rich dynamic visuals.

System requirements

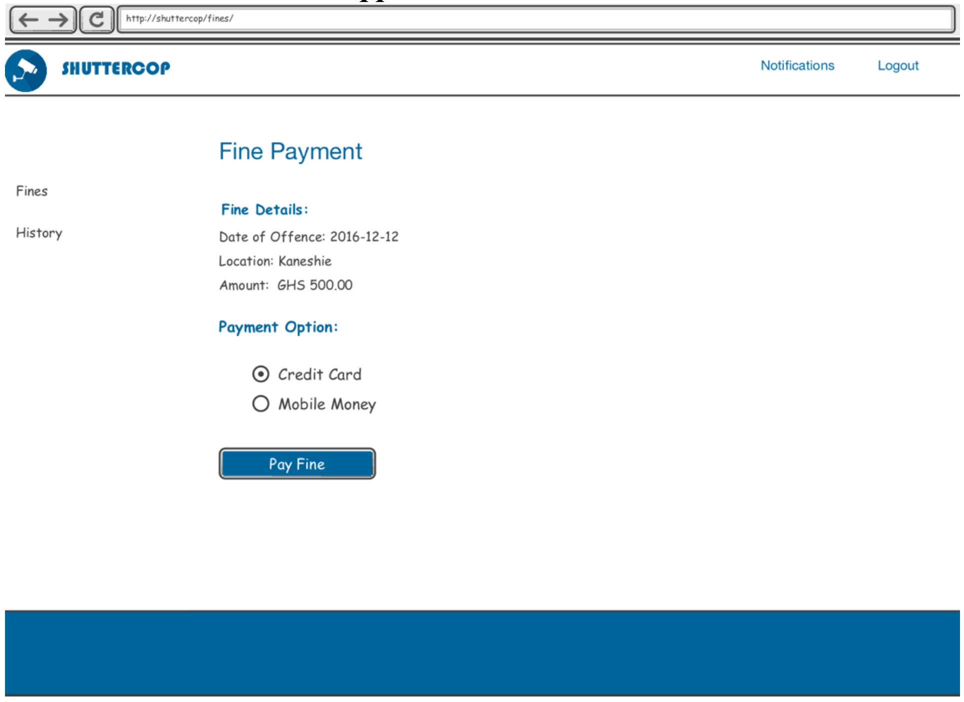
- Encrypted secret and key authentication: The application must have a secure and encrypted authentication system for users of the API.

3.4 External Interface Requirements

3.4.1 User Interface

The mockup user interface for the driver and law enforcer client side applications are shown in Figures 3.2 and 3.3 respectively. The law enforcer's page is also shown in Figure 3.4.

3.4.1.1 Drivers client side application



Fine Payment

Fine Details:

Date of Offence: 2016-12-12

Location: Kaneshie

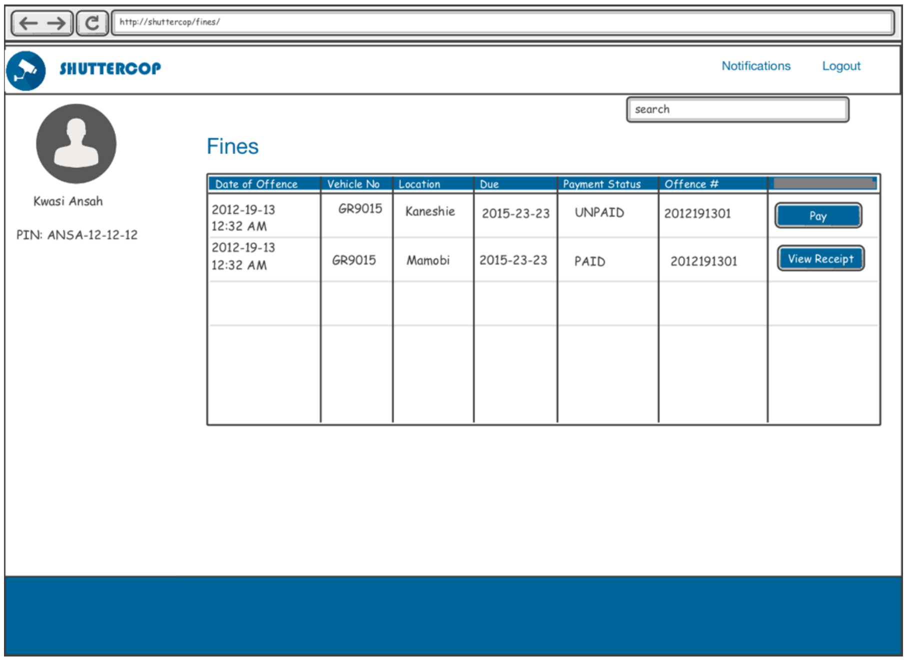
Amount: GHS 500.00

Payment Option:

[✓ Credit Card]

Mobile Money

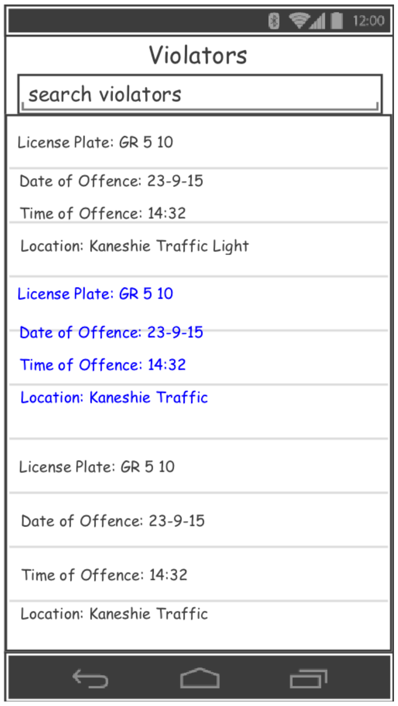
Figure 3.2: Driver's fine payment portal user interface



Date of Offence	Vehicle No	Location	Due	Payment Status	Offence #	
2012-19-13 12:32 AM	GR9015	Kaneshie	2015-23-23	UNPAID	2012191301	Pay
2012-19-13 12:32 AM	GR9015	Mamobi	2015-23-23	PAID	2012191301	View Receipt

Figure 3.3: Driver's fines and offences view page

3.4.1.2 Law Enforcer's client side application



License Plate: GR 5 10
Date of Offence: 23-9-15
Time of Offence: 14:32
Location: Kaneshie Traffic Light

License Plate: GR 5 10
Date of Offence: 23-9-15
Time of Offence: 14:32
Location: Kaneshie Traffic

License Plate: GR 5 10
Date of Offence: 23-9-15
Time of Offence: 14:32
Location: Kaneshie Traffic

Figure 3.4: Law enforcer's view page

3.4.2 Hardware Interfaces

ShutterCop is intended as graphics-intensive server-side application and hence will be hosted on server which has the required memory and processing power to quickly process the images. The ShutterCop server will also handle payment transactions, message notifications and data exchange between client-side devices such as Android devices or phones that are not smart. Information will be sent using TCP/IP protocols. The ShutterCop Image capture camera will transmit images to the ShutterCop server using TCP/IP protocols.

3.4.3 Software Interfaces

ShutterCop system will be developed with three languages: Python, PHP and/or Android. The image processing component of the ShutterCop Server will be developed with OpenCV libraries and Tesseract OCR Engine. PHP/Python will be used to display user pages and handle payment transactions and Android will be used to develop the client-side user interface for users who have an Android device.

3.4.3.1 Incoming and outgoing items

Incoming data consists of captured images, and payment transactions sent by traffic post cameras and users to the server.

Outgoing data consists of email and SMS notifications sent by the server to users regarding their fine amount or confirmation of fine payment.

3.4.3.2 Services and communications

ShutterCop relies on server push and pull protocols in order to be fully functional. Communication will be event-based and will occur between the traffic post cameras, the image processing server and the user’s phone or email in the following situations:

Whenever the traffic cameras capture an image

- Whenever the server successfully extracts a number plate
- The server will notify users when that they have been fined
- Whenever a user confirms a transaction to pay their fine

3.4.4 Communications Interfaces

ShutterCop system will have a network server that is web-based and created using Python (for image processing) and PHP (for HTTP protocols). The server exists to process images, retrieve information from the driver and vehicle database, make notifications and confirm payments. The system requires a database for storing driver information about users, their vehicles and payment transactions in paying their fines. The HTTP server will use a push protocol to push notifications to both Android and non-Android phones.

3.5 Other Nonfunctional Requirements

3.5.1 Performance Requirements

Performance is critical for the system to work well since high resolution images will be transmitted to the server. Moreover, the image processing algorithm requires a lot of computational power in order to process images quickly.

3.5.2 Security Requirements

The software assumes that users will have access to a handheld device or a computer. Hence, the major security concerns are securing the email addresses, phone numbers, confidential driver information. Moreover, because monetary transactions will be made through the system, Https will be used to secure drivers' confidential financial information from hackers.

3.6 Summary of Requirements Specifications

In this chapter, requirements specifications have been discussed. This contained a high level overview of the system architecture, user requirements and system requirements for implementing the ShutterCop ALPR system. In the next chapter, the methodology and design specification are discussed.

Chapter 4: Methodology & Design Specification

This chapter describes the methodology that will be used in developing the proposed system in Section 4.1. Section 4.2 gives a detailed description of the design specifications of the system as well as the architecture used to develop the system.

4.1 Methodology

In Ghana, there are essentially two kinds of license plates: non-commercial license plates, which have black characters on a white plate and commercial license plates which have black characters on a yellow plate. This system focuses exclusively on the former category of license plates. A high-level overview of the system is shown in the flow diagram in Figure. 4.1. The components and structure of the proposed system are described in this section. The waterfall model will be used in developing this system. Consequently, all requirements analysis and design specification will be done before the implementation and testing of the system.

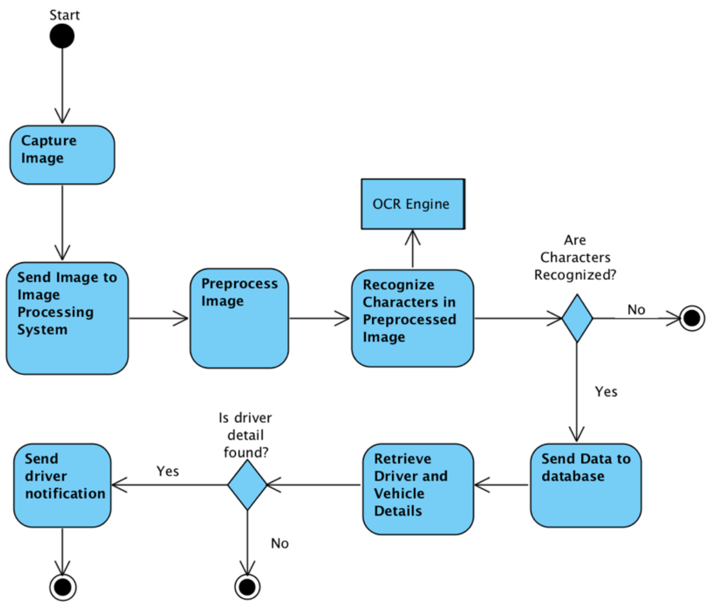

Start
Capture Image
Send Image to Image Processing System
Preprocess Image
OCR Engine
Recognize Characters in Preprocessed Image
Are Characters Recognized?
No
Yes
Send Data to database
Retrieve Driver and Vehicle Details
Is driver detail found?
Yes
No
Send driver notification

Figure. 4.1: Shows the activities involved in the systems

4.2 Image Capture

In lieu of a high resolution and high speed shutter camera, a phone camera with very good resolution (greater than 300dpi) will be used. Ideally, an infrared camera would have been best suited for this system. Moreover, OCR engines are very sensitive to skew in images, hence, the angle and positioning of the camera is essential in increasing the success ratio of the system. In the proposed system, slanted images will not be used.

4.3 Image transfer to OCR system

Captured images transferred from an Android phone will interface with an apache server and sent to a python socket server as a base 64 string.

4.4 Image Preprocessing

Image Preprocessing is significant in increasing the accuracy of character recognition algorithms or engines. For license plate recognition, this comprises a set of algorithms that will be applied on images to enhance quality. Essential preprocessing operations and algorithms include resizing, noise reduction, grayscale conversion, binarization and thresholding, edge detection and character segmentation. These preprocessing algorithms and operations are discussed in subsequent sections.

4.4.1 Resizing

Images captured by the cameras may be too large in size. Hence resizing images is necessary to increase performance of the system and save disk space.

4.4.2 Noise Reduction and Image Smoothing

It is only natural that images captured contain significant noise and characters that are irrelevant to the system but could significantly decrease the accuracy of the character recognition system. In order to minimize the effect of noise on the accuracy of the system, input images will have to be corrected for noise. Morphological operations will be used to reduce the effect of noise.

4.4.3 Grayscale Conversion

Input images must be converted to grayscale. Captured images will normally be in RGB mode comprising three channels (viz. red, green and blue). This implies extra colour information for each channel. Images have to be converted to a single grayscale channel to get rid of extraneous colour information. Grayscale conversion results in an image of 256 grey levels, which is an essential precondition for binarization and thresholding.

4.4.4 Binarization and Thresholding:

Binarization and thresholding is the process of converting image pixels into black or white values using a threshold. OCR engines have a higher accuracy when input images are binarized (i.e. converted to black and white). Thresholding can either be global or adaptive. With global thresholding gray image pixel values are truncated to just two values depending on a fixed threshold value. On the other hand, with adaptive thresholding pixel values are truncated based on the mean threshold value of neighbouring pixels. For a more agile and adaptive system, the adaptive thresholding techniques will be used.

4.4.5 Edge Detection and Localization of Number Plate

Not everything in the image is of interest to the system. Edge detection algorithms such as Sobel Edge or the Canny Edge operator will be applied to identify the edges and contours of the number plate and isolate it from the rest of the image.

4.5 Character Segmentation

Character segmentation is the process of cropping out and segmenting characters of interest for identification. Segments will then be sent for identification. Unrecognized character segments will be discarded.

4.6 Character Recognition

After sufficient preprocessing, selected character segments are fed to the OCR engine for identification which with success will return the license number.

4.7 Retrieval of Information from Database

After successful identification of a license number, the license number will be used to retrieve information about the owner or driver of the vehicle.

4.8 Notification of Involved and Interested Persons

Notification of drivers or vehicle owners will be done via SMS to reach a wider pool of persons of interest that may not own a smart phone.

4.9 Design Specification

This section provides a comprehensive overview of the ShutterCop ALPR System. It represents a selection of different architectural views that show different parts of the system. It is intended to capture and convey relevant architectural decisions that have been made on the system. It is based on the "4+1" model view of architecture. Based on this model the views represented are (i) the logical view, (ii) the implementation view, (iii) the process view, (iv) physical view and (v) the use case view.

4.9.1 Logical View

This subsection describes the functionality that the proposed system presents to end-users. A sequence diagram is used to present this view to end users. As illustrated in Figure 4.2, the image capture system is triggered to capture the image of a vehicle, it then sends the image to the image processing server. After successful recognition of the vehicle license number, the driver details associated with the license number are retrieved from the database. If a record is found the driver is sent a notification.

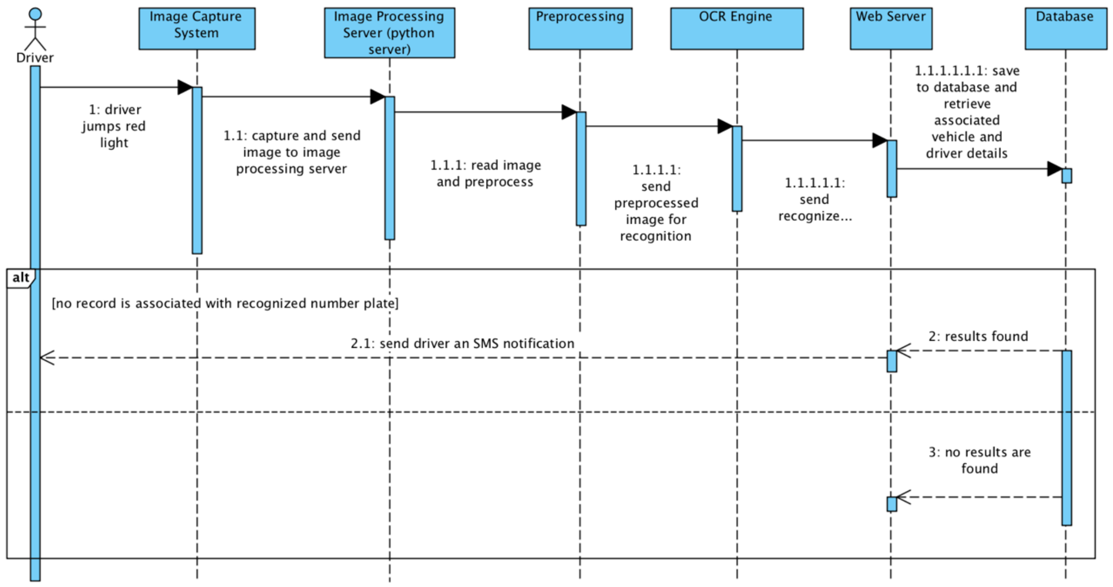

Driver
Image Capture System
Image Processing Server (python server)
Preprocessing
OCR Engine
Web Server
Database
1: driver jumps red light
1.1: capture and send image to image processing server
1.1.1: read image and preprocess
1.1.1.1: send preprocessed image for recognition
1.1.1.1.1: send recognize...
1.1.1.1.1.1: save to database and retrieve associated vehicle and driver details
alt
[no record is associated with recognized number plate]
2.1: send driver an SMS notification
2: results found
3: no results are found

Figure 4.2: Shows a sequence of activities and a timeline of events that occur when the system is triggered

4.9.2 Process View

This view describes the design's concurrency and synchronization aspects. An activity diagram is used to illustrate this view.

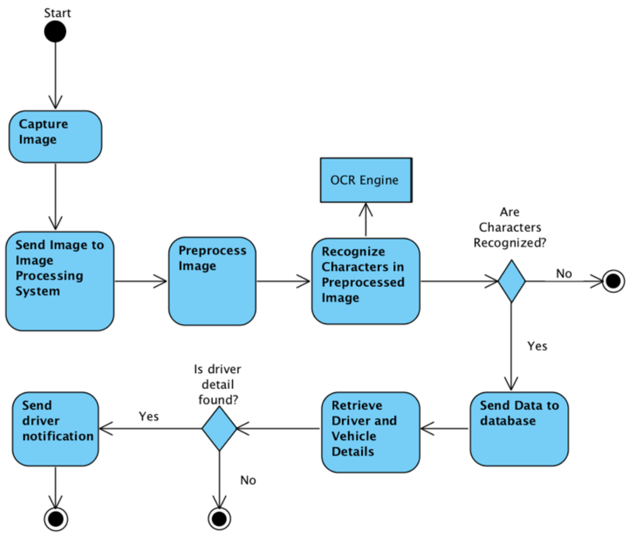

Start
Capture Image
Send Image to Image Processing System
Preprocess Image
Recognize Characters in Preprocessed Image
OCR Engine
Are Characters Recognized?
No
Yes
Send Data to database
Retrieve Driver and Vehicle Details
Is driver detail found?
Yes
No
Send driver notification

Figure 4.3: Shows the activities involved from image capturing to notification of driver

4.9.3 Physical View

This view depicts the ShutterCop ALPR system from the engineer's perspective. It describes the mapping of the software onto the hardware and shows the system's distributed aspects. A deployment diagram is used to show this aspect of the system.

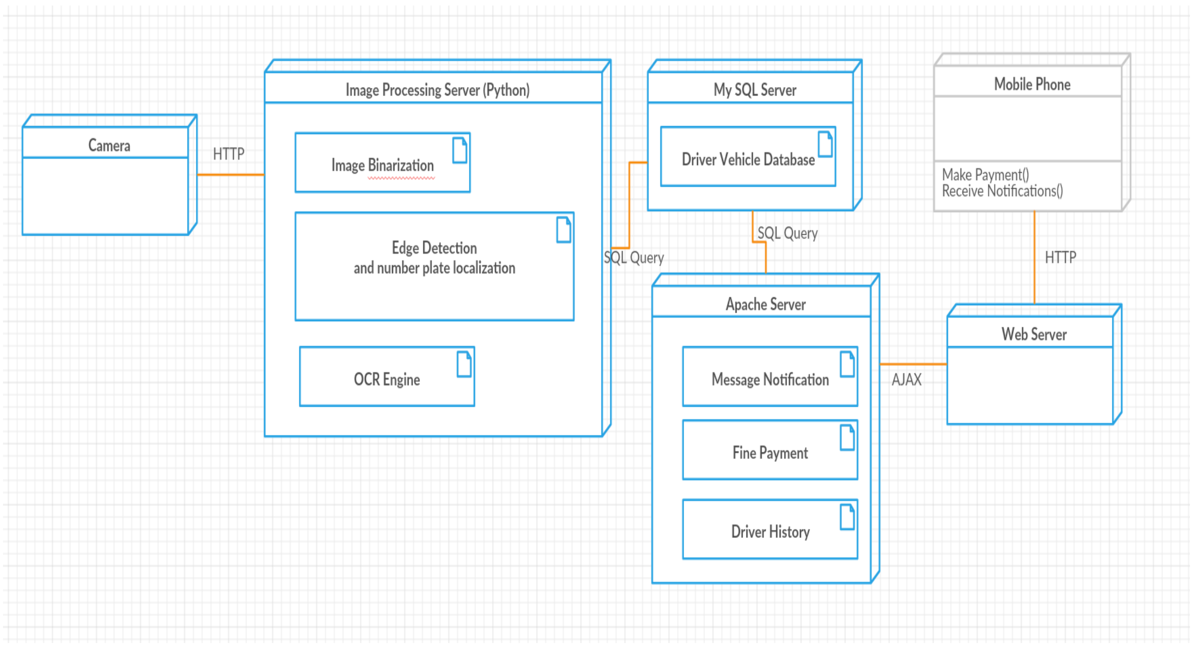


Figure. 4.4: Shows a mapping of software onto the hardware in the system and how they are distributed

4.9.4 Use Case View

This view describes all the stakeholders that will be involved in the proposed system as well as the functionalities that are required by them. A use-case diagram is used to show this view in the diagram below.

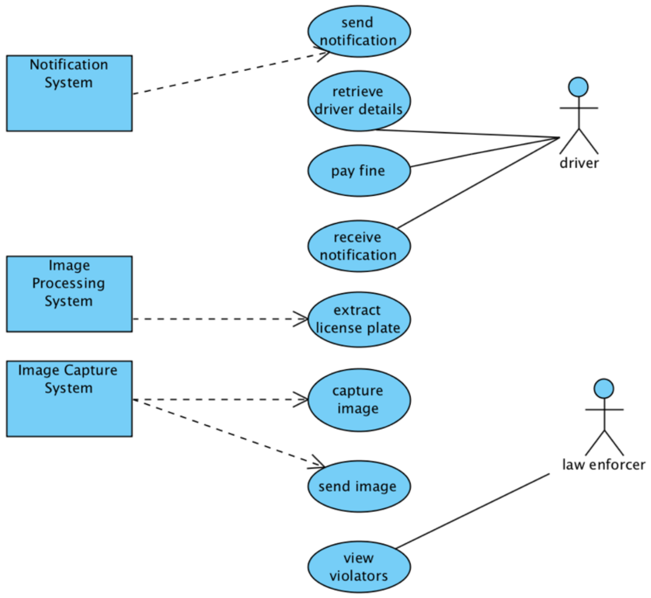


Figure 4.5: Shows usage scenarios of identified stakeholders of the system

4.9.5 Data Model

This section describes the data model that will be used to keep records of details and activities in the ShutterCop ALPR system. An Entity Relation Diagram is used to describe the architecturally significant persistent elements in the data model.

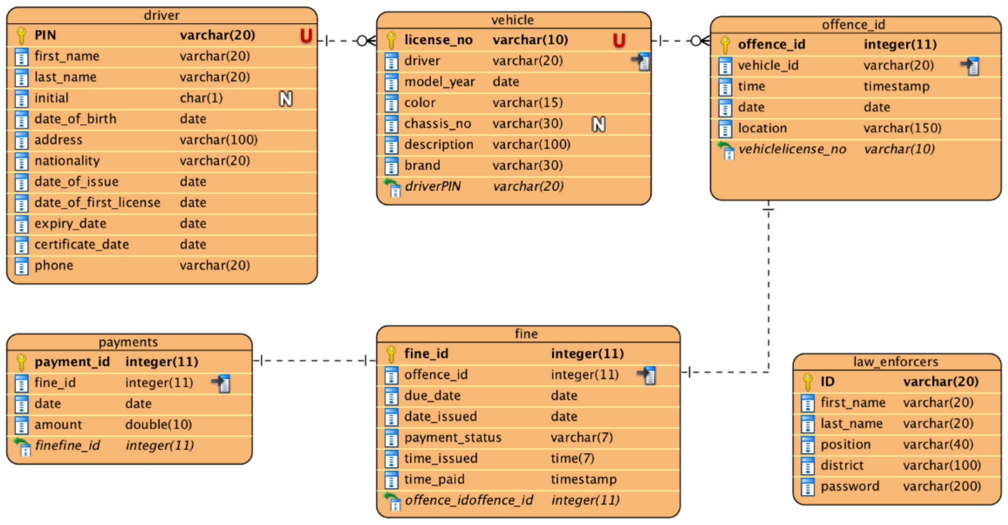


Figure 4.6: Shows an entity relation diagram describing the data model for the client side of the proposed system

4.10 Summary of Methodology and Design Specification

In this chapter the methodology and architecture governing the implementation of this system has been discussed. As mentioned in section 4.1, the waterfall model will be used in developing the ShutterCop ALPR system. Additionally, with the use of appropriate UML diagrams, the architectural design of the proposed system has been specified. In the next chapter, the implementation process is described in detail.

Chapter 5: Implementation

5.1 Implementation

This section describes in detail the processes involved in implementing the ShutterCop ALPR system. In the subsequent subsections all of the essential procedures, algorithms and snippets of code that were used to implement the system are detailed. It starts with discussing the technique for simulating the traffic scenario and capturing input images, and moves onto discussing how the images were transferred to the image processing engine. It then describes how images were preprocessed, segmented and fed into the OCR engine for recognition. Finally, it describes how persons of interest are notified via SMS and how the RESTFUL insurance API was implemented.

5.2 Image Capture and Traffic Light Simulation

The traffic light scenario was simulated with an Arduino board connected to an HC-05 Bluetooth module and three light emitting diodes (LED): red, green and yellow. The setup is shown in Figure 5.1. The Bluetooth module was used to establish a serial connection with a Samsung Galaxy S3 android phone and send data to the phone to capture images when the red LED comes on. To establish communication with the phone, an Android application was developed that utilized the phone's Bluetooth communication serial port to send and receive data from the Bluetooth module on the Arduino board. Code snippets showing how the Bluetooth system was implemented in Arduino and Android are shown in Appendix A and Appendix B respectively. Upon receiving the data from the Bluetooth module, images were captured with an 8 Mega Pixel (MP) Samsung Galaxy S3 phone camera.

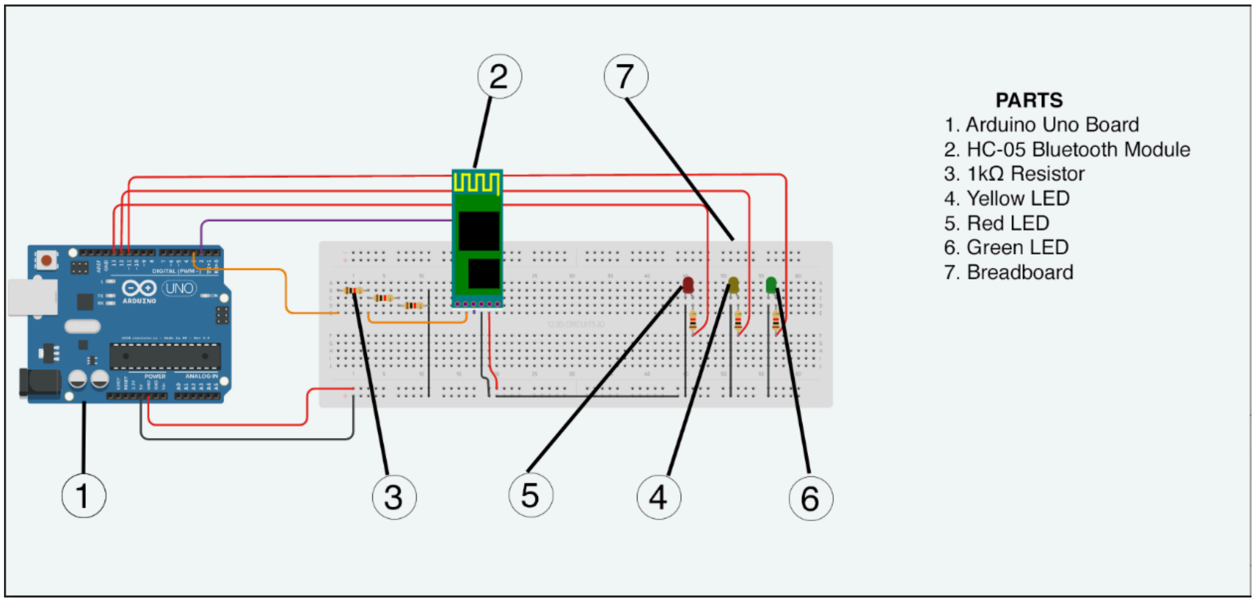


Figure 5.1: Traffic light simulation setup with Arduino UNO, Bluetooth module and LEDs

5.3 Image Transfer

The captured images were transferred from an Android phone to an intermediate PHP server as a base 64 string and relayed to the Python server via socket communication.

5.4 Image Preprocessing

Image preprocessing is the most essential step in building an ANPR system. It is a combination of image processing algorithms crucial to determining and extracting the region of interest in captured images and removing noise so that characters can be segmented and recognized with a high degree of accuracy. The image preprocessing techniques used in developing the ShutterCop ANPR system are described in detail in the subsections of this section.

5.4.1 Image Resizing

The first preprocessing step that was employed was resizing input images to standard sizes. This step was necessitated by the fact that input images that were too large hence, consumed to much disk space and significantly reduced the speed and efficiency other image processing algorithms. However, since good image quality is also a requirement for improved accuracy of the OCR engine, only images whose dimensions exceeded 1000 by 1000 were resized. The implementation of this function is shown in appendix C. of the appendix below. It employs the OpenCV resize function in resizing images to desired sizes.

5.4.2 Grayscale Conversion

A prerequisite for binarization and thresholding is that input images have to be grayscale. This was achieved with the OPENCV function cvtColor illustrated in appendix C.

5.4.3 Binarization and Thresholding

In this phase, thresholding and binarization were performed to convert grayscale images to just black and white pixel images. Prior research indicated that an adaptive thresholding algorithm was required to maintain image quality. In implementing this feature, Otsu binarization and simple thresholding algorithms were combined to obtain a much better output. OpenCV provides an implementation of this combination of thresholding algorithms which is in appendix C.

5.4.4 Morphological Operations

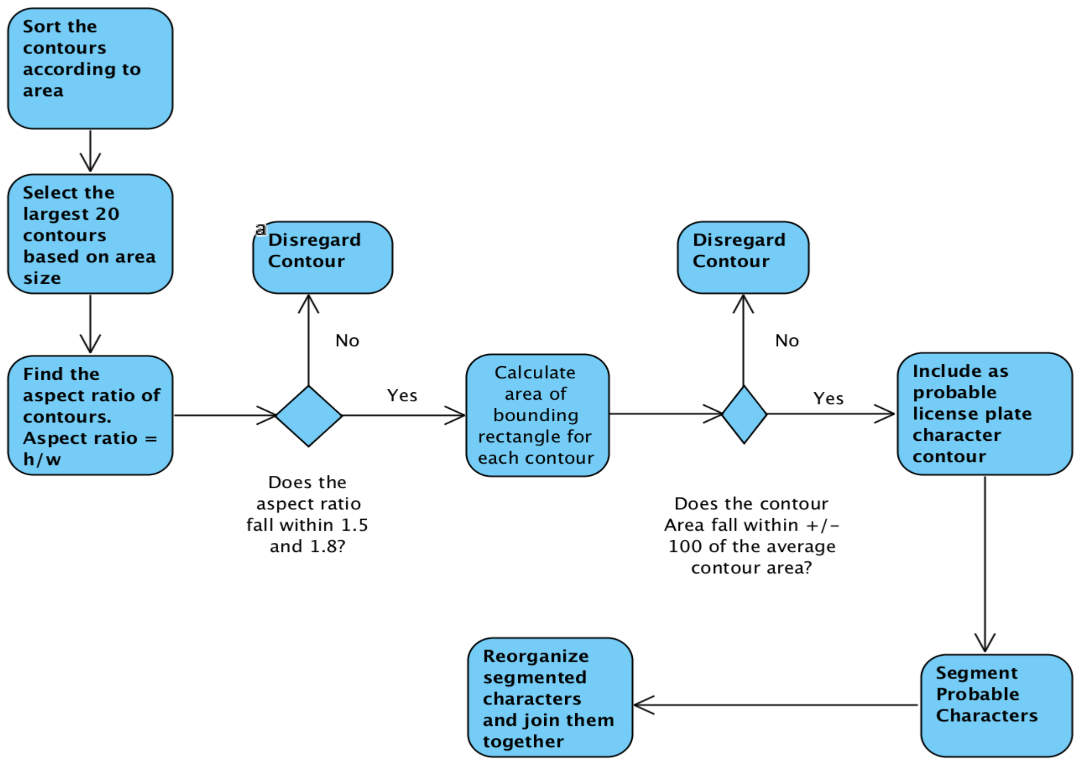
After an adaptive thresholding algorithm was applied to the image, a series of morphological operations were applied to further outline character edges and minimize noise. To do this, the binarized image was eroded and subsequently dilated. The code for implementing this operation is shown in appendix C.

5.4.5 Edge Detection and Number Plate Localization

An edge detection algorithm was then applied to the resulting image from the previous step. Canny edge detection algorithm was used to identify edges in this resulting image. Canny edge detection algorithm is a combination of Gaussian blur algorithm and Sobel Edge detection algorithm. OpenCV provides an implementation of this algorithm which is utilized in the development of this system. After the edges had been detected with the Canny Edge operation, contours were identified and filtered according to size and shape to obtain the region that had the highest probability of containing the number plate. After a successful identification of the number plate area, the region of interest was cropped out for further processing. The implementation for these processes is indicated in appendix C of the appendix.

5.4.6 Character Segmentation

After the region containing the number plate had been successfully localized or identified, probable number plate characters were identified and segmented by contour area and aspect ratio. Contours were filtered according to aspect ratio based on the observation that the average aspect ratios of probable number plate characters fall between 1.5 and 1.8. Subsequently, segmented characters were sorted and stacked together to obtain the desired image for OCR engine. The algorithm used to segment and reorganize characters is shown in Figure 5.2 and 5.3 respectively. Additionally, code snippets of these algorithms are shown in appendix D.



Sort the contours according to area
Select the largest 20 contours based on area size
Find the aspect ratio of contours. Aspect ratio = h/w
Does the aspect ratio fall within 1.5 and 1.8?
No
a
Disregard Contour
Yes
Calculate area of bounding rectangle for each contour
Does the contour Area fall within +/- 100 of the average contour area?
No
Disregard Contour
Yes
Include as probable license plate character contour
Segment Probable Characters
Reorganize segmented characters and join them together

Figure 5.2: Contour filtering and segmentation algorithm

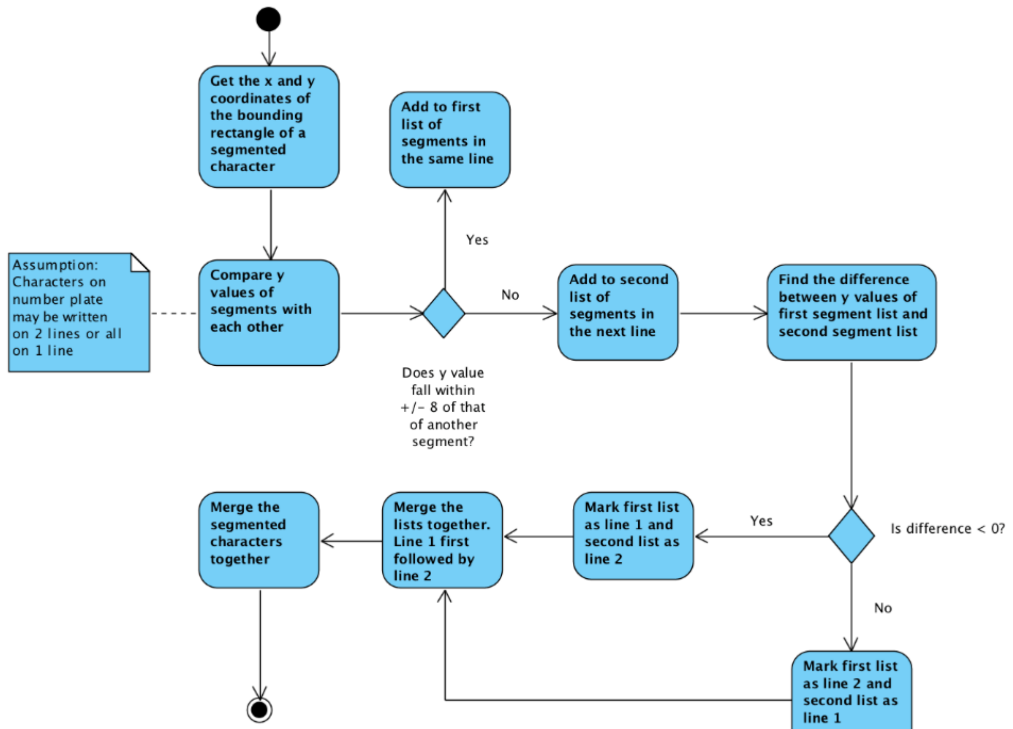

Get the x and y coordinates of the bounding rectangle of a segmented character
Add to first list of segments in the same line
Yes
Assumption: Characters on number plate may be written on 2 lines or all on 1 line
Compare y values of segments with each other
No
Add to second list of segments in the next line
Find the difference between y values of first segment list and second segment list
Does y value fall within +/- 8 of that of another segment?
Merge the segmented characters together
Merge the lists together. Line 1 first followed by line 2
Mark first list as line 1 and second list as line 2
Yes
Is difference < 0?
No
Mark first list as line 2 and second list as line 1

Figure 5.3: Shows the algorithm used to implement the character reorganization

5.5 Character Recognition

Following a successful character segmentation, the stacked image was fed to the Tesseract OCR Engine for character recognition. In order to eliminate unlikely number plate characters, a reference document containing only candidate characters for was created and used to match characters recognized by the Tesseract OCR engine. This was done via commands evoked in python shown below:

```
os.system('tesseract /path/to/image <output_file>')
```

5.6 SMS Notification

Following a successful recognition of the license plate, the corresponding driver's details were retrieved from the database and used to send the driver an SMS notification. The SMSGH API was used to send SMS notifications through the SMSGH gateways to drivers.

5.7 RESTFUL API for Insurance companies

In implementing the restful API, a non repeating random character generator function was written to generate unique access credential. Following this, the generated credentials were encrypted with a secure password hashing function and stored in the database. An API user authentication class was developed to handle user connection to the system. Code listing showing how this functionality was implemented is shown in Appendix E.

5.8 Summary of Implementation

In this chapter, the processes used to implement the ShutterCop ALPR system have been discussed. Also, how the techniques discussed in chapters 2 and 4 were implemented have been more technically explained. In the next chapter, the procedures and criteria for testing the implemented system are discussed in detail. The results of the tests conducted are also discussed.

Chapter 6: Testing & Results

6.1 Testing

In this section, testing procedures used to assess the performance of the various subsystems of the ShutterCop ALPR system are described.

6.1.1 Test Description

Since the system comprises many parts, it was necessary to test each subsystem or unit appropriately. The client side application of the system was tested with various unit tests and component tests. PHP unit tests were written to test functions on the web application. The image processing subsystem was tested by inputting a number of images into the script to determine how well the characters in images were recognized. The image processing subsystem was tested with the following criteria:

- How well it was able to detect and localize a number plate in a given image
- How well it was able to identify probable number plate characters and segment them
- The accuracy of recognized characters compared with actual characters on the number plate

6.1.2 Unit Tests

PHP unit tests were written to test the functions and classes for the client side applications for drivers and law enforcers. One of these unit test classes is shown in code listing in Appendix F.

6.1.3 Component Testing

Component tests were conducted on the image processing system with a variety of images to test the robustness and reliability of various components. Three main components of the image processing system were tested. These include the number plate localization module, the character segmentation module and finally the character recognition module. The number plate localization or identification module was tested with a number of images with varying brightness and noise intensity and varying distance at which the images were taken from vehicle. Additionally, images taken at night were compared with those taken at day to find out how successful the system was at locating number plate in images under contrasting lighting conditions. The character segmentation module was tested by varying parameters used for the reducing noise and identifying edges in the image to determine what parameters increased the likelihood of identifying probable characters for segmentation. Finally, the character recognition module was tested with images of varying resolution, noise intensity and tilt to determine the acceptable levels of tilt, noise and resolution required for successful recognition of characters.

6.1.4 System Testing

The system was tested by varying these parameters: distance from the vehicle to the phone camera and lighting conditions under which the image was taken. Distances at which images were captured were varied by 30 cm starting from 60 cm from the vehicle so as to determine the range at which the recognition system was still effective. For this, a total of 39 images taken at 30cm intervals from vehicles were used.

With regard to testing by varying lighting conditions, the image dataset was categorized according to when the image was taken. One set of images comprised number plates taken at night and the other set consisted of license plate images taken during the day. The images used for testing were taken with an 8 mega-pixel Samsung Galaxy S3 phone camera. Furthermore, the average execution time of the system was calculated to determine how fast the system processed images. The results of the system testing are discussed in the next section.

6.2 Results

The objectives of this project as outlined in section 1.5 of the introductory chapter have to a substantial extent been achieved. It is however, worth mentioning to what extent the set objectives have been realized. The application to transfer images to the image processing server has been implemented. Secondly, the system to notify drivers of their offence and the associated fines has also been implemented. More importantly, the ALPR system has also been implemented with somewhat satisfactory levels of accuracy across the aforementioned testing parameters and criteria discussed in the previous section on testing. The results obtained from testing all these subsystems are detailed in subsequent paragraphs.

First, after testing the image processing and OCR system on 39 images, the system recorded and average execution time of 0.17 seconds. The accuracy of the image processing system was measured with respect to the three modules mentioned in section 6.1.1. These are the (i) number plate localization or detection module, (ii) the character segmentation module and (iii) the character recognition module. Across all testing parameters, the number plate localization/detection module recorded a 61.5 % success rate, implying that for 25 out of the 39 test images, the system was able to accurately detect the location of a number plate in the image. In Figure 6.1 and Figure 6.2 images showing a successful detection of a number plate is compared with another in which the system failed to accurately detect a number plate. The character segmentation and reorganization module recorded an accuracy rate of 77% for all test images used. Contrasting images showing when the system successfully segmented probable number plate characters are shown in Figure 6.3 and 6.4. This shows that the system's character reorganization algorithm, detailed in section 5.4.6 and Figure 5.3, works fine for both single row number plates and two row number plates. However, character segmentation is not very successful when some characters are joined together as it is in Figure 6.3. This may occur due to the presence of fastening bolts and nuts in number plates. Still across all testing parameters, the character recognition module produced an accuracy rate of 63% for all input images that contained number plates and were successfully detected and localized.

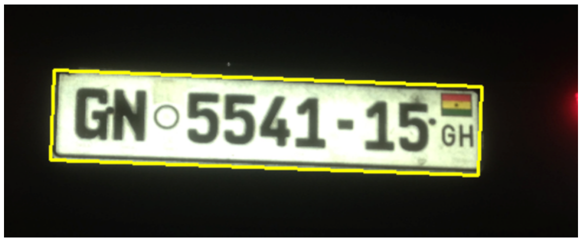


Fig. 6.1: Successful detection of number plate

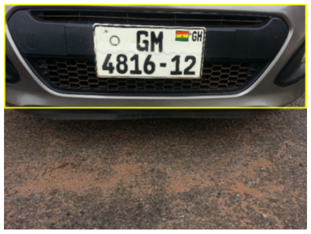


Fig. 6.2: Unsuccessful detection of license plate

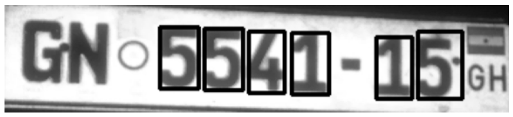


Fig.6.3: Successfully segmented characters

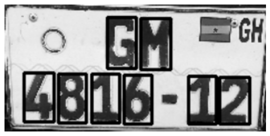


Fig. 6.4: Successful segmentation of characters

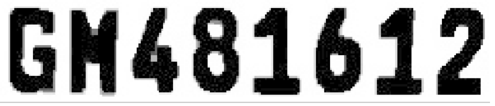


Fig 6.5: Successful reorganization of segmented characters on a two row number plate

Additionally, the results from testing the image processing system at various distances from the vehicle are shown in Table 6.1. From the table it was identified that images taken at distances further than 150 cm from the vehicle had the lowest accuracy with regard to all the three modules. The system was able to identify, segment and recognize characters in an image with a higher degree of accuracy if the image was taken at a distance no greater than 120cm from the vehicle.

The typical distance of a CCTV camera is 500-3000cm. These cameras are however fitted with lenses so images can be zoomed in and are much clearer.

Table 6.1: Accuracy rates of image processing modules at different intervals/distances

Distance From Vehicle/cm	Number Plate Detection (success Rate)	Accuracy of Segmentation	Accuracy of Recognized Characters
60	75%	68%	63%
90	75%	65%	51%
120	71%	58%	49%
150	57%	50%	41%
180	43%	14%	5%
210	33%	0%	0%

Further tests conducted with regard to lighting conditions indicated that the image processing system was comparably more accurate during the day when there was more light than during the night (Table 6.2). This result could be heavily influenced by the type of camera used because most images taken at night were quite blurry because the number plate and reflectors reflected the flash back into the camera. It must be noted that these images were also taken at distances between 60cm and 180cm.

Table 6.2: Accuracy rate of image processing modules under different lighting conditions

Time of Day	Number Plate Detection (success Rate)	Accuracy of Segmentation	Accuracy of Recognized Characters
Day	61.5%	77%	63%
Night	60%	31%	23%

Altogether, all components of the ShutterCop ALPR system have been implemented to a satisfactory level. However, as indicated from the results discussed in this section, more work is required to improve the system.

6.3 Summary of Testing & Results

In this chapter, the testing procedures and criteria have been explained. Results obtained from testing the implemented system have also been discussed in detail. In the next chapter, limitations of the implemented solution are discussed and further work to improve the system is suggested.

Chapter 7: Conclusion, Limitations & Recommendations

7.1 Limitations

As mentioned in the introduction, the accuracy of ALPR systems is critically affected by conditions such as lighting, resolution, tilt and noise. The ShutterCop ALPR systems also suffers from the same limitations even though it does present its benefits. First of all, the system is seriously affected by lighting conditions. It has been identified that the system works better for images taken during the day than those taken at night. This limits the effectiveness of the system since quite a number of drivers drive at night.

Another limitation of the system is that it cannot accurately recognize characters in slanted images. The more slanted the image is, the less likely the system is to recognize the characters. Moreover, results gathered through testing show that number plate detection module is not as robust as it should be. In very noisy images, the localization module is less accurate.
In addition, a phone camera was used to take the images, this limits the quality of images taken. Better results could be achieved if an infrared camera is used instead of the phone camera. Furthermore, the system was limited to only non-commercial vehicles because such number plates were not readily available for testing. This cuts out a significant portion of motorists on Ghanaian roads.

7.2 Future Work

Based on the limitations discussed earlier, there is a lot more that can be done to improve the current system to reach industry standards. The localization module can be improved to attain the same degree of accuracy for images taken in the day as it does for images taken at night. Moreover, an algorithm can be developed to detect and straighten slanted images so that they can also be recognized. Also, the system should extend its coverage to commercial vehicles as well to increase its effect. Furthermore, a payment module, such as Mobile Money can be integrated to enable drivers pay fines without having to drive to the police station. Finally, a much better camera, preferably an infrared camera, can be used to obtain better images.

7.3 Conclusion

Inasmuch as vehicular transport has become an inextricable part of our daily lives, it is also responsible for staggering amounts of fatalities. Several measures have been taken to minimize the carnage caused by road accidents in Ghana. Still roads remain unsafe. Authorities and traffic law enforcers have gradually begun to adopt technological solutions in an attempt to minimize road collisions as utilized in advanced countries. One such technology, used in developed countries to make roads safer, is the ALPR. However, because of the costs associated with installing these systems, they have not been adopted yet. This project identified various technologies that can be exploited to develop a cost effective ALPR system for Ghana roads. The technologies that were employed include SMS platforms, computer vision or image processing platforms, OCR libraries and robust programming languages.

The implemented system integrated the SMS GH gateway, the Tesseract OCR engine, OpenCV and NumPy image processing libraries. Additionally, a traffic light scenario was simulated with an Arduino board. The system can identify non-commercial vehicle license plates in an image and extract text likely to be a license number with fairly high degree of accuracy. Following a successful extraction of characters in a number plate, the system is able to notify appropriate stakeholders that they have been identified.

With the use of a combination of free open source software, this project has indicated that cost effective ALPR systems can be developed to make Ghanaian roads safer.

7.4 Business Model

Beyond the technical component extensively elucidated in previous chapters, the ShutterCop ALPR system can be run as a business. As a business, the system presently targets two kinds of customers: the police force and insurance companies. Law enforcement could utilize the system to easily collect fines and keep track of driver history. In addition, the system could be employed by insurance companies to inform decisions concerning whether or not to increase insurance premium for a driver based on their driving history.

In terms of revenue generation, a mixed transaction model will be used. First of all, every correctly recognized license plate, will mean that the ShutterCop account will be credited with 5% of fine amount. Insurance companies, however, will be charged a yearly subscription fee for access to the API.

References

Brahmbhatt, S. (2013). *Practical OpenCV.* New York: Springer.

Duan, T. D., Duc, D. A., & Du, T. L. (2004). Combining Hough tranform and contour algorithm for detecting vehicles' license plates. *Intelligent Symposium on Intelligent Multimedia, Video and Speech Processing* (pp. 747-750). Hong Kong: IEEE.

FrontlineSMS. (2016). *FrontlineSMS Overview*. Retrieved March 28, 2016, from FrontlineSMS: http://www.frontlinesms.com/technologies/frontlinesms-overview/

Gaumont, N., & Babineau, D. (2008, November 11). *The role of automatic license plate recognition technology in policing: results from Lower Mainlanf of British Columbia.* Retrieved March 26, 2016, from The Police Chief: http://www.policechiefmagazine.org/magazine/index.cfm?fuseaction=display&article_id=1671

Goyal, M. (2011). Morphological image processing. *International Journal of Computer Science & Technology , 2* (4), 161-165.

Huang, Y., Lai, S., & Chuang, W. (2004). A template-based model for license plate recognition. *Networking, Sensing and Control, 2004 IEEE International Conference. 2*, pp. 737-742. IEEE.

Kasaei, S. H., & Kasaei, S. M. (2011). Extraction and recognition of the vehicle license plate for passing under outside environment. *Intelligence and Security Informatics Conference (EISIC), 2011 European* (pp. 234-237). Athens: IEEE.

Kim, K. K., Kim, K. I., Kim, J. B., & Kim, H. J. (2000). Learning-based approach for license plate recognition. *Neural Networks for Signal Processing X, 2000. Proceedings of the 2000 IEEE Signal Processing Society Workshop. 2*, pp. 614-623. IEEE.

Mahalakshmi, T., Muthaiah, R., & Swaminathan, P. (2012). Review article: an overview of template matching technique in image processing. *Research Journal of Applied Sciences, Engineering and Technology , 4* (24), 5469-5473.

Martinsky, O. (2007). *Algorithmic and mathematical principles of automatic number plate recognition systems.* Brno University of Technology, Department of Intelligent Systems. Brno: BRNO University of Technology.

Mordvintsev, A., & Abid, K. (2013). *Canny Edge Detection*. Retrieved 2 16, 2016, from OpenCV-Python Tutorials: https://opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html#canny

Neal, & Kevin. (2010, January 13). *Why is OCR at 300 dpi a standard?* Retrieved April 4, 2016, from Scan Snap Community: http://scansnapcommunity.com/tips-tricks/1652-why-is-ocr- at-300-dpi-a-standard/

NumPy. (2016). *NumPy*. Retrieved March 28, 2016, from NumPy.org: http://www.numpy.org

Ocropy. (2016). *OCRopus*. Retrieved March 28, 2016, from Ocropy: https://github.com/tmbdev/ocropy

OpenALPR. (2016). *OpenALPR Cloud API*. Retrieved March 28, 2016, from OpenALPR.com: https://github.com/openalpr/openalpr

Oppong, R. A. (2012). *Statistical Analysis of Road Accidents Fatality In Ghana Using Poisson Regression.* Kwame Nkrumah University of Science and Technology , Department of Mathematics . Kumasi: College of Science.

Rosebrock, A. (2014, January 12). *My Top 9 Favorite Python Libraries for Building Image Search Engines.* Retrieved March 28, 2016, from PyImageSearch: http://www.pyimagesearch.com/2014/01/12/my-top-9-favorite-python-libraries-for-building-image-search-engines/

Rouse, M. (2008, November). *Android OS*. Retrieved March 28, 2016, from SearchEnterpriseLinux: http://searchenterpriselinux.techtarget.com/definition/Android

Sauvola, J., & PietikaKinen, M. (2000). Adaptive document image binarization. *The Journal of Pattern Recognition , 33*, 225-236.

SciPy. (2016). *Scikit-Image* . Retrieved March 28, 2016, from Scikit-Image.org: http://scikit-image.org/docs/dev/

Shrivastava, V., & Sharma, S. (2012). Artificial neural networks based optical character recognition. *Signal & Image Processing : An International Journal , 3* (5), 73-80.

Solanki, R., Rai, R., & Teena, R. (2013). The automatic license plate pecognition (ALPR). *International Journal of Research in Engineering and Technology , II* (7), 161-167.

Tesseract OCR. (2016). *External tools, wrappers and training projects for Tesseract.* Retrieved March 28, 2016, from Tesseract: https://github.com/tesseract-ocr/tesseract/wiki/AddOns#tesseract-wrappers

World Health Organization. (2015). *Global status report on road safety.* Geneva.

Yanamura, Y., Goto, M., Nishiyama, D., Soga, M., Nakatani, H., & Saji, H. (2003). Extraction and tracking of number plate using Hough transform and voted block matching. *Intelligent Vehicles Symposium* (pp. 243-246). IEEE.

Yoshimori, S., Mitsukura, Y., Fukumi, M., & Akamatsu, N. (2003). License plate detection system in rainy days. *Computational Intelligence in Robotics and Automation, 2003. Proceedings. 2003 IEEE International Symposium. 2*, pp. 972 - 976. IEEE.

Zhai, X., Bensaali, F., & Sotudeh, R. (2012). OCR-based neural network for ANPR. *Imaging Systems and Techniques (IST), 2012 IEEE International Conference* (pp. 393-397). IEEE.

Appendices

Appendix A: Arduino Traffic Light Simulation Code Snippet

```
#include <SoftwareSerial.h>

SoftwareSerial mySerial(2, 3);
int dataFromBT;
int red = 13; // red led PIN
int yellow = 12;  // yellow led PIN
int green = 11;  // green led PIN

void setup() {
  Serial.begin(9600);
  Serial.println("LED simulation starting");

  mySerial.begin(9600);
  pinMode(red,OUTPUT);
  pinMode(yellow,OUTPUT);
  pinMode(green,OUTPUT);
}

void loop() {

  mySerial.println(" ON ");
  delay(5000);

  if (mySerial.available()){
    dataFromBT = mySerial.read();
  }

  if (dataFromBT == '0') {
    // Turn off LED
    digitalWrite(13, LOW);
    Serial.println(dataFromBT);
    mySerial.println(dataFromBT);

  } else if (dataFromBT == '1') {
    // Turn on LED
    trafficLights();

  }
}

//function for traffic light simulation
void trafficLight(){
  digitalWrite(red, LOW);
  digitalWrite(yellow, LOW);
  digitalWrite(green, HIGH);
  delay(5000);
  digitalWrite(red, LOW);
  digitalWrite(yellow, HIGH);
  digitalWrite(green, LOW);
  delay(5000);

  digitalWrite(red, HIGH);
  digitalWrite(yellow, LOW);
  digitalWrite(green, HIGH);
  delay(5000);

  mySerial.println(" ON ");
}
```

Appendix B: Android Bluetooth Connection Code Listing

```
private class ConnectToBluetooth extends AsyncTask<Void, Void, Void>
{

    private boolean isConnected = true;

    @Override
    protected void onPreExecute()
    {
        progressDialog = ProgressDialog.show(Simulation.this,
getString(R.string.connecting), getString(R.string.wait));
    }

    @Override
    protected Void doInBackground(Void... devices)
    {
        try
        {
            if (bluetoothSocket == null || !isBtConnected)
            {
                //get the default device's bluetooth adapter
                bluetoothAdapter =
BluetoothAdapter.getDefaultAdapter();

                //get connection to bluetooth module
                BluetoothDevice bluetoothModule =
bluetoothAdapter.getRemoteDevice(deviceMAC);

                // create a Serial Port Profile (RFCOMM) connection
                bluetoothSocket =
bluetoothModule.createInsecureRfcommSocketToServiceRecord(myUUID);


BluetoothAdapter.getDefaultAdapter().cancelDiscovery();

                //start connection to bluetooth module
                bluetoothSocket.connect();

                //get output stream to bluetooth
                outStream = bluetoothSocket.getOutputStream();

                //get input stream to bluetooth
                in = bluetoothSocket.getInputStream();
            }
        }
        catch (IOException e)
        {
            isConnected = false;//if the try failed, you can check the
exception here
        }
        return null;
    }
    @Override
    protected void onPostExecute(Void result) //after the
doInBackground, it checks if everything went fine
    {
        super.onPostExecute(result);

        if (!isConnected)
        {
            Toast.makeText(getApplicationContext(),
R.string.btConnFailed, Toast.LENGTH_LONG).show();
            response.append(getString(R.string.txtViewBtConnFailed));
            finish();
        }
        else
        {
            Toast.makeText(getApplicationContext(),
R.string.connected, Toast.LENGTH_LONG).show();
            response.append(getString(R.string.connected) +
getString(R.string.breakln));
            isBtConnected = true;

            //start thread to read from bluetooth device
            bluetoothReceiver = new BluetoothReceiver(bluetoothSocket,
response);
            bluetoothReceiver.start();
        }
        progressDialog.dismiss();
    }
}
```

Appendix C: Procedures for the image processing and recognition

This section describes the procedures used to process and recognize images containing number plates.

Required Packages

```
#import necessary packages
import numpy as np
import cv2
from matplotlib import pyplot as plt
from PIL import Image
import os
import Reordering
```

Image Resize Algorithm

```
#resize image if the dimensions exceed 1000 by 1000
def resizeInputImg(image):
    if width >= 1000 and height >= 1000:
     #use opencv resize function
        return image
    else:
        return image
```

Grayscale Conversion

```
#3. convert to grayscale
img = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
```

Binarization and Thresholding

```
#9. binarize cropped image with simple thresholding and otsu

ret,inv_thresh = cv2.threshold(cropped_img, 127, 255 ,
cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
```

Morphological Operations

```
#use opening to remove unwanted white spaces or noise
opening_kernel = np.ones((3,3),np.uint8)
img = cv2.morphologyEx(img, cv2.MORPH_OPEN, opening_kernel)
```

Edge Detection

```
#11. detect edges in the image

edges = cv2.Canny(img,100,200)
```

Appendix D: Algorithm for Segmenting & Reordering Characters

This module presents the essential parts of the algorithm implemented to segment characters and reorder them.

Character Segmentation Algorithm

```
#This function filters contours by size and aspect ratio

def isProbableCharacter(x,y,w,h):

        #get aspect ratio of character

        #get area of character

        #filter according to aspect ratio and area
        if ratio >= 1.5 and ratio <= 1.8:
            if area > 50:
                return True
        else:
            return False
```

Character Reordering Algorithm

```
#this function identifies characters that are on the same line or
row and groups them

def rowSort(matrix):

    #define acceptable threshold for y values difference
    #declare list for all rows
    #initialize list for first row
    #initialize list for second row

    for i in range(0, len(matrix)): #iterate over matrix of contours
        if y values fall within the acceptable threshold and is not in
the
            and first row list
            #append to the first row
        else if it is not in the second row:
            #append to second row list
    #append all rows to the list for all rows and return it
```

```
#this function orders contours in the same row'''

def orderRows(row, all_rows):
    ordered_row = []
    #sort the contours in the given list in descending order by x
values
    ordered_row = sorted(ordered_row, key=lambda x: x[0]) #order
contours by


# this function determines which segments are on top of each other
based on
y coordinates

def isOnTop(all_rows, matrix):
    combined_rows = []

    # if the length of the row is greater than 2
    if len(all_rows) >= 2 and len(all_rows[1]) > 0:
        first_row_y = all_rows[0][0] #get y-value of contour in the
first row
        second_row_y = all_rows[1][0] #get y-value of contour in the
second row

        #if first row y is greater than second row y
        #then characters in the first row are on top of the second row
        if matrix[first_row_y][1] < matrix[second_row_y][1]:

            #order x values of first row
            first_row = orderRows(all_rows[0], matrix)

            #then order x values of second rows
            second_row = orderRows(all_rows[1], matrix)

            #join first row and second row together
            combined_rows.extend(first_row)
            combined_rows.extend(second_row)

            return combined_rows
        else:
        #characters in the second row are on top of the first row

            #order first row and second row
            first_row = orderRows(all_rows[1], matrix)
            second_row = orderRows(all_rows[0], matrix)

            #join second row first and then join first row
            combined_rows.extend(second_row)
            combined_rows.extend(first_row)

            return combined_rows
    else:
        #the number plate is a single row number plate
```

```
#order first row
first_row = orderRows(all_rows[0], matrix)
combined_rows.extend(first_row)

return
return combined_rows
```

Appendix E: API User Authentication Class Code

```
class AuthApiUser extends adb_object{

    var $secret ;
    var $key;

    /**
     * AuthApiUser constructor.
     */
    function __construct(){
        parent:: __construct();
    }

    /**
     * @return bool|mysqli_stmt
     */
    function addNewUser(){
        $str_query = "INSERT INTO api_users(user_key, secret)
                      VALUES(?,?)";

        $stmt = $this->prepareQuery($str_query);

        if($stmt === false){
            return false;
        }

        $this->secret = assignSecret();
        $this->key = assignKey();

        $enc_secret = encrypt($this->secret);
        $enc_key = $this->key;

        $stmt->bind_param("ss", $enc_key, $enc_secret);

        $stmt->execute();

        return $stmt;
    }

    /**
     * @param $secret
     * @param $key
     * @return bool
     */
    function authenticateUser($secret, $key){

        $result = $this->getUser($key);
        $row = $result->fetch_assoc();

        if(count($row) == 0){
            return false;
        }else{
            $enc_key = $row['secret'];
            return verifyKey($secret, $enc_key);
        }
    }

    /**
     * @param $user
     * @return bool|mysqli_result
     */
    private function getUser($user){
        $str_query = "SELECT AU.secret, AU.user_key
                      FROM api_users AU
                      WHERE AU.user_key = ?";

        $stmt = $this->prepareQuery($str_query);

        if($stmt === false){
            return false;
        }

        $stmt->bind_param("s", $user);

        $stmt->execute();

        return $stmt->get_result();
    }

    /**
     * @return mixed
     */
    function returnSecret(){
        return $this->secret;
    }

    /**
     * @return mixed
     */
    function returnKey(){
        return $this->key;
    }

}
```

Appendix F: Unit Test class for the Offences

```
class OffenceTest extends PHPUnit_Framework_TestCase
{


    public function testAddOffence(){

        $testObj = new Offence();
        $vehicle_no = 'GR111111';
        $location = 'Test Location';
        $this->assertType(mysqli_stmt,
$testObj->addOffence($vehicle_no, $location));

    }


    public function testGetDriverOffence(){
        $testObj = new Offence();
        $driver = 'XXXXXXXXXXXXX';
        $this->assertType(mysqli_result,
$testObj->getDriverOffence($driver));
    }


    public function testGetNumDriverOffences(){
        $testObj = new Offence();
        $driver = 'XXXXXXXXXXXXX';
        $this->assertType(mysqli_result,
$testObj->getNumDriverOffences($driver));
    }

}
```