Adama Science and Technology University

School of Electrical Engineering and computing

Department of Computer Science and Engineering

A SENIOR PROJECT DOCUMENTATION SUBMITTED FOR THE PARTIAL FULFILLMENT OF DEGREE PROGRAM AT DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Project Title: Real-time vehicle overspeed detector and notifier

Group members

Amanuel Girma	**A/UR5051/09**
Amhir Edris	**A/UR4168/09**
Lencha Fikiru	**A/UR4355/09**
Meheratu Tamiru	**A/UR4727/09**
Mekuanent Molla	**A/UR3939/09**

Supervised by: Dr. Biruk Yirga
August 25, 2021
Adama, Ethiopia

Submitted by

Amanuel Girma	______________________	Aug 25, 2021
Student	Signature	Date
Amhir Edris	______________________	Aug 25, 2021
Student	Signature	Date
Lencha Fikiru	______________________	Aug 25, 2021
Student	Signature	Date
Meheratu Tamiru	______________________	Aug 25, 2021
Student	Signature	Date
Mekuanenet Molla	______________________	Aug 25, 2021
Student	Signature	Date

Approved by

1. Dr. Biruk Yirga	______________________	Aug 25, 2021
Advisor	Signature	Date
2. ______________________	______________________	______________________
Chairman, Dept.'s Senior project Committee Department	Signature	Date
3. ______________________	______________________	______________________
Head of the Dept.	Signature	Date

ACKNOWLEDGEMENT

First of all, we would like to extend our deepest gratitude to the unpaid ever help of the Almighty God starting from the beginning to the completion of our project. It is all His work to have the courage and persistency for achieving once dream. We have no words to thank Him enough for all His grace, guidance and unlimited pleasures and blessings.

Secondly, we would like to express our appreciation for Adama Science and Technology University specially our computer science and engineering department for creating a peaceful learning environment that allowed us to work on this project. We have become familiar with different tools and techniques during the making of this project.

Thirdly, we would like to say thank you to our advisor Mrs. Yordanos Gebeyehu for all the support she provided us during the first stage development of this project. She has been so helpful and supportive to us.

Last but not least, our special sincere thanks go to our advisor, Dr. Biruk Yirga., CSE department head who guided us in the second phase development of this project, for his patient and honest guidance and advice that enabled us to keep our strength and hope to finish this project. His kindness and concern to provide all the possible helps he can offer to us was equipping us with motivation and determination to keep working hard towards the completion of our project. We would like to say thank you to our advisor for all his kind and patient guidance and advice.

Table of Contents

List of figures

List of tables

ACRONYM

A

ASTU- Adama Science and Technology University

B

BR- business rule

C

CSS- Cascading Style Sheet

CSE- Computer Science and Engineering

G

GSM- Global System for Mobile Communication

GPS- Global Positioning System

H

HTML- Hypertext Markup Language

L

LCD- Liquid Crystal Display

LPR- License Plate Recognition

P

PL- presentation layer

W

WYSIWYG- what you see is what you get

ABSTRACAT

In Ethiopia, there is still a high rate of road accidents. Most of these accidents are caused by over speeding. Most of the drivers are reckless and do not obey the speed set for given road places. On how to cub this, currently traffic polices around Addis Ababa, measure over speeding using speed Guns. The speed Guns does not measure over speeding but rather calculate the speed at which the vehicle is travelling and require human intervention to detect over speeding.

This project is then all about how to detect speed limit violation and notify it to relevant authorities in real-time. This is achieved using GPS module that collect the GPS data and GSM module that sends over speed alarm to the EVSC system and then to the nearest traffic police via EVSC app. This helps in the advancement of science and technology by diversifying the usage of GSM, GPS and GPRS technologies.

CHAPTER 1

INTRODUCTION

1.1 Overview

Despite the measurements taken, car accident is still the number one cause of young people’s deaths worldwide. We are losing millions of working forces daily. Peoples are dying so frequently by car accidents everywhere. In most regions of the world this epidemic of road traffic injuries is still increasing. Our country Ethiopia, is one of those countries who reports car accidents daily. When we watch news every time, we hear a news of people’s deaths by car accident. Even this year (2013 E.C) we, ASTU students, were horrified by the deaths of one of our universities graduated student and a second-year student by car accident.

The main cause of most of car’s accident is over speeding. Many drivers do not respect the speed limit posed by road traffic authorities. They over speed when they are at the location where there are no traffic police.

This project will contribute to the reduction of Ethiopia’s car accident problems that happens due to over speeding. We are proposing a way to track and log an over speeding car by using GSM and GPS and notify it to the closest traffic police using an android app, to reduce these car accidents that happens due to over speeding. We will develop a controller that is attached to vehicles body and keeps track of places and time the vehicle over speed and send the event information to the central system.

1.2 Background of the project

The design and development idea of this real time Vehicle over speed detector and notifier System come from irresponsible and reckless speeding of some vehicles such as truck, trailer, lorries and buses, which have resulted in fatal accidents leading to loss of innocent road users lives and fatal body damage. There are some factors that lead to these road accidents, such as vehicle overload, atmospheric condition as well as speeding. Since these commercial vehicles have poor braking capability and greater momentum at higher speed, speeding will be considered as a base for the warning system development. Besides that, with large amount of the commercial vehicles operating in country, it is virtually impossible for the authorities to control all these drivers continuously.

There are a number of out-of-vehicle and in-vehicle speed tracking and monitoring technologies with the potential to enhance speed compliance. Since the out-of-vehicle speed tracking device will only make the drivers obey the speed limit just in the speed enforcement zone, our System will focus on the in-vehicle approaches, which will continuously monitor these vehicle speeds at all time and will record the relevant information.

The real time Vehicle over speed detector and notifier system consists of three primary subsystems. These are: - the in-vehicle subsystem, the central control subsystem and android application for traffic police. The in-vehicle subsystem is a main control module that will be installed in the vehicle for speed monitoring, where the date, time and total time for speeding when the over speeding happened will be recorded into the memory and displayed on LCD display, whereas the central monitoring subsystem is a supporting system to interface and communicate with the in-vehicle subsystem by using wireless communication module, including data setting and retrieving. Once the speeding records in the in-vehicle subsystem memory has been retrieved by the wireless communication module, a web system can be used to display and analyze the data, providing variable and useful insight into the vehicle's speeding information. The other part of real time Vehicle over speed detector and notifier System is the part that notifies committed speed limit violations to the traffic police in line with the position where the vehicle is headed. It is an android app that receives data from the central control system. It notifies to the traffic polices that, a vehicle with mentioned plate number has committed a speed limit violation and is headed towards him/her. Based on this, Traffic Polices working on the field can enforce legal action on the over speeding vehicle.

1.3 Statement of the problem

In most countries a number of ways are being implemented to check and identify the over speeding vehicle. But no centralized system has been developed so far that can perform the task of speed detection and notification without human assistance.

Here in Ethiopia, the current existing system, to tackle this life-threatening problem is assigning traffic polices daily to control the road traffic. This traffic polices sometimes, use sophisticated radar or Li-DAR guns to determine how fast a vehicle is moving. Both of these devices are used to identify distance and speed. In other words, they determine how far away an object is or how fast they are moving. These instruments have a capacity of measuring speed and distance only at the time of measuring. Traffic polices do not have a means of finding out by what speed the car has been moving before that. These devices do not tell them the exact location where the driver passed the allowed speed limit.

And another problem here is that, many of these road traffic polices are corrupted. They will take money and let the vehicle pass even though the law is broken. Ethiopia does not have a centralized real time road control system. It is hard to know where and when the vehicle had passed the speed limit and for that, whether that vehicle driver is fined or not. Generally, the major issues seen in the existing systems are:

- Road accidents are increasing day by day with prime cause being the over speeding of the vehicle.
- The use of human resources to check this issue can be very tedious and time consuming and sometimes become irrelevant.
- Commercially available automatic LPR systems are very costly and difficult to implement.

1.4 Objective of the project

1.4.1 General objective

The general objective of our project is to design vehicle speed detector that detect the over speeding event of vehicles and developing centralized controlling system with android mobile application notifies to traffic police.

1.4.2 Specific objective

In order to attain the general objective, we set the following list of specific objectives:

- ✓ Study and analyze the existing vehicle speed detection and controlling systems and technologies.
- ✓ Design vehicle speed detector(controller) which installed on a vehicle that senses the over speed event and send the sensed event to the central monitoring system.
- ✓ Displaying from information's sent to EVSC system databases to the vehicle driver on the LCD display.
- ✓ Designing and developing a web-based system that, receives information from controller and store on the central database, display received information from the database to the authorized unit, send notification to the traffic police.
- ✓ Developing EVSC app that connects traffic polices to EVSC system in order to receive notifications from EVSC system and send report of actions taken back to EVSC system.
- ✓ Allowing Traffic police to report the action taken to centralized control system using android app.
- ✓ Implement the system using the selected tools.
- ✓ Test and deploy the system.

1.5 Feasibility study

1.5.1 Financial feasibility

Our System include web application that need to be hosted which require hosting costs. Since the System transfer data only in text form, the bandwidth required for the operation of this application is very low. Each and every vehicle owner should have to purchase and configure the designed hardware part that is going to communicate with the central system and android app. The System will follow freeware software standards.

No cost will be charged from the potential customers once system setup and vehicle configuration are completed. Bug Fixes and maintenance will have an associated cost beside the associated costs. Our system benefits the customer in different ways for instance every vehicle speed can be tracked wherever it is and nearby Traffic Police Can be notified if any over speed is encountered. From these it is clear that the project is Financially Feasible.

1.5.2 Technical feasibility

The Project involve Android based and Web based. The main technologies and Tools that we are going to use will be: HTML, CSS, java script framework(react) for front end, python frame work (Django) for back-end development, MYSQL database, Android Studio, GSM enabled devices, GPS enabled devices, Arduino, LCD displays. Each of the above technologies are available on the market and our team members acquire required skill to work with them. Having this in our mind we can conclude that our project is Technically Feasible.

1.5.3 Operational feasibility

The Proposed Project is operationally feasible for sure because it's solution maker for the current critical problem of world society. The Project is going to present easy and flexible environment to the end user, when the project is ready for Production any one that has contact to the System can access it without any difficulties.

1.6 Scope and Limitation

1.6.1 Scope of the project

- ✓ It detects the speed of vehicles and incase over-speeding occurs, notifies it to the EVSC system
- ✓ EVSC app for traffic polices that helps them receive notifications from EVSC system to report the action taken.
- ✓ Register and manage Vehicles and traffic polices.
- ✓ It works in Adama city.

1.6.2 Limitation

Although computerized systems are advantageous regarding the simplicity of the task, the reliability won't be like that from human work. At such, our system has different limitations:

- ✓ The speed limit seated for the vehicles is variable depending on the location of the vehicle which is not provided by Google's Google Map for our country Ethiopia. This limitation will force us to assume and give general speed limits for different location manually by ourselves which is an overhead.
- ✓ This system, even though motor cycles are included under the Vehicle categories, it does not detect motor cycles over speeding.
- ✓ The device that is going to be configured on the vehicles have to gather and send the required data from the vehicle to the central database often for further computation but to perform all this tasks it needs to be online and functional but there are no ways to detect if the driver unplugs the system from the vehicle.
- ✓ GPS Accuracy depends on a number of variables, most notably signal to noise ratio (noisy reception), satellite position, weather and obstructions such as buildings and mountains. These factors can create errors in our perceived location and speed measurements.
- ✓ Due to time constraints, we haven't included many features that can be configured with our systems like detecting occurrence of accidents, alarm for driver.
- ✓ Both EVSC system and EVSC app functionality depends on the availability of internet connection. If there is no internet connection the system will not work.

1.7 Significance of the project

- ✓ In its long term this system will decrease the number of vehicle accidents that happens due to over speeding.
- ✓ This system will keep track of vehicles that has committed the speed limit violation and what kind of punishments they have been punished.
- ✓ It is well suitable for all vehicles and it does not need any human intervention.
- ✓ Possible to track the location of every vehicle.

1.8 Beneficiaries of the project

The project's result could be applicable in different areas benefiting different target groups. The main beneficiaries could be the following.

- ✓ Ministry of Transport- to identify and monitor registered vehicles.
- ✓ Peoples using road transport will be saved from the car accidents that is caused by over-speeding problems.
- ✓ Pedesterians

1. 9 Methodology

1.9.1 Data collection methodology

For the purpose of requirement elicitation for the new system, primary data from the federal police commission and other potential organizations such as minister of transport and Ethiopia road authority for their accident data recording and reporting requirements, has been collected and analyzed. Vehicle over-speed detecting and monitoring designed and implemented in other countries have been also consulted for incorporating key and useful design and implementation features with the current system.

1.9.2 System development methodology

Among the existing software development process models, we use Agile Model to develop our system. Because agile method is an iterative, team-based, communicative with the users, time-boxed approach development nature. Agile software development and testing follow a process that helps teams deliver a working product that provides value at the end of each sprint. Embracing change is one of the core tenets of the process. With Agile software development process, we can quickly adapt to requirement changes without negatively impacting release dates. Not only that, Agile helps us to reduce technical debt, improve customer satisfaction and deliver a higher quality product.

1.10 Development tool

1.10.1 Hardware tools

- ❖ **GPS Module:** It provides multiple and accurate data items, including the speed, location, date and time.

- ❖ **GSM module:** used to enable communication between a microcontroller (or Arduino Uno) and the GSM / GPSR Network.
- ❖ **Micro-controller**: is a small computer on a single integrated circuit and used in automatically controlled products and devices.
- ❖ **LCD display:** It is one kind of electronic display module
- ❖ **Power supply**

1.10.2 Software tools

- ❖ **Android Studio:** It will be used to develop the mobile application that the traffic police going to use.
- ❖ **Google maps:** The Google Maps Road Apps permits to plot GPS coordinates to geometry of the road; it is also used to identify the vehicle's speed limits on the road segments. The following are services exposed for Google Maps Apps:
  - ✓ Snap to Roads: It returns the best-fit geometry of the road for a provided GPS coordinates set.
  - ✓ Nearest Roads: It returns individual road divisions for a provided GPS coordinates set.
  - ✓ Speed Limits: It returns the positioned speed limit for the road segment.
- ❖ **Django:** A python programming language-based web framework, which helps to connect back end and front end together.
- ❖ **HTML, CSS, JavaScript**
- ❖ **Arduino Uno IDE:** we use it to configure the components connected to the Arduino Uno module.
- ❖ **MS-Word:** we use it for document preparation.
- ❖ **XAMPP:** XAMPP was used to create a local web server for testing and deployment purposes (Local Hosting).
- ❖ **Arduino IDE**
- ❖ **Proteus Suit**

1.11 Testing Plan

1.11.1 Unit plan

At this phase of testing, we will test the individual units/ components of a software to validate that each unit of the software performs as designed and eliminate faults in procedure and functions point of view by using black box and white box testing.

Tasks we will perform under this phase are:

- ✓ Unit test plan
- ✓ Identify the unit test objectives

- ✓ Prepare test cases that includes information such as set of test inputs, execution condition and expected output
- ✓ Perform the tests according to our plan
- ✓ Analyze the test results
- ✓ Document the test results

1.11.2 Integration test

At this phase of testing, we will combine the individual units and taste them as a group and this is performed after the unit testing.

The following tasks are going to be performed under this phase: -

- ✓ Prepare integration test plans
- ✓ Identify integration test objectives
- ✓ Identify integration test acceptance criteria
- ✓ Perform the tests according to our plan
- ✓ Document the test results

1.11.3 System and acceptance test

System and acceptance test testing would also be part of this testing phase to determine if the product as a whole performs as needed and that it can be accepted by the test model organization Adama Traffic and Road Authority.

1.12 Required resource with cost

Materials Needed	**Amount**	**Price of each in birr**	**Total cost in birr**
GPS Modem	1	550	1200
GSM Modem	1	100	500
LCD display	1	200	200
Arduino Uno	1	900	900
Circuitry devices	-	1500	1500
others	-	1000	1000
Total			5800

Table 1. 1 required resource and cost

1.13 Task and schedule

	Task / Activity	Progress (in %)	Due Date	Source of Insight/Support (References)
1	Identifying the problem	100	2021/02/18	Ministry of Transport's reports
2	Prioritize problems	100	2021/02/18	observation
2	Gathering information (Look into previously made related researches on the IoT based framework for vehicle Over-speed detection)	9o	2021/02/18	Google Scholars https://www.researchget.net. ERA/ MoT
3	Analyze information	75	2021/02/20	Project Team
4	Evaluate Ministry of Transport's reports on the issue of over speeding and death	100	2021/02/20	Ministry of Transport's Website http://www.era.gov.et
5	Looking into the working principle of Arduino, GPS and GSM technologies and how they can be interface using Arduino	10	2021/02/21	https://www.researchget.net https://www.google.com
6	Designing		2021/03/25	Project Team
7	Implementation and Integration	0	2021/05/13	Project Team
8	Testing	0	2021/07/23	Project Team

Table 1. 2 task and schedule

1.14 Team composition role

	NAME	ID NO	ROLL IN TEAM
1	Amhir Edris	A/UR4168/09	Back-end and Android
2	Mekuanent Molla	A/UR3939/09	Back-end and Front-end
3	Amanuel Girma	A/UR5051/09	Android and Hardware
4	Lencha Fikiru	A/UR4355/09	Front-end and Android/Team Leader
5	Meheratu Tamiru	A/UR4727/09	Hardware and Front-end

Table 1. 3 team composition and role

CHAPTER TWO

DESCRIPTION OF THE EXISTING SYSTEM

The current systems that are being used by traffic police is to control over speeding are out-vehicle technology devices that will only make the drivers obey the speed limit just in the speed enforcement zone like a radar speed gun. A radar speed gun is a device used to measure the speed of moving objects. It is used in law-enforcement to measure the speed of moving vehicles from a distance.

2.1 Major function of the current system

Police use sophisticated radar or LiDAR guns to determine how fast a vehicle is moving. Both of these devices are used to identify distance and speed. In other words, they determine how far away an object is or how fast they are moving.

To determine the speed of moving objects, a radar gun will send several signals toward a moving vehicle. As the vehicle moves toward the radar gun, the return signal has a shorter distance to travel. The radar device will use this change in frequency to determine the speed of the car.

2.2 Users of current system

The users of the current system are:

- ✓ **Traffic polices**: - each traffic polices use the speed limiter system which is installed on cars system and use manual and printed traffic rules and regulation of traffic out of Addis Ababa and after that take action on drives that obey the rule and regulation.
- ✓ **Federal transport bureau**: -by using the system speed limiters they assist the traffic services to the city dwellers. And if there is over speeding driver, they punish as the rule. example if driver drives above 80km/hr. in Addis Ababa, he is guilty and fined by low. Traffic polices out of Addis Ababa uses the current manual and printed system to control and manage traffic issues that can be handle in every region of the country.

2.3 Drawback off current system

Currently there is no way to detect over speeding of all the vehicles available throughout the country. Tools such as Li-DAR, which is very expensive and handset tools traffic police use to measure over speed at a specific location and time. This means that other times and locations, where traffic policies cannot reach, there exists no method or techniques of over speeding detection and control.

The other thing is the current systems' approach to pedestrian safety illustrates a problematic focus of the "National Road Safety Strategic Plan of Ethiopia 2011 to 2021" (NRSSP) on perfecting humans, rather than protecting humans. It focuses almost entirely on motorist and pedestrian awareness behavior, and should be realigned to best practice approaches to pedestrian safety, which can be address by building systems that detect and control over speeding vehicles like the one we are proposing to build, as outlined in the latest World Health Organization manual for pedestrian safety.

There are also issues concerning data capturing and management techniques. Issues like corruption in traffic polices and drivers, inefficiency in management systems, loss of documents and so on are the biggest headaches of the existing system.

2.4 Business Rule

BR-01: A traffic controller shall request and take driving qualification license from a vehicle driver who violates the provisions of traffic rules and give appropriate charge letter which commands to pay fines.

BR-02: A traffic controller or other legally authorized person shall give appropriate charge letter to a pedestrian or any other person who violates the provisions of this regulation; the detail implementation of which shall be determined by the directive to be issued by the Authority.

BR:03 Any driver fined, shall pay the penalty within 10 working days, unless he reverses his/her penalty through complaint lodged.

BR-04: If a driver fined does not pay the penalty within 10 working days, he/she shall be subject to additional penalty and to be recorded in punishment list. With regard to person other than driver, similar compliance shall be determined by a directive to be issued by the authority.

BR-05: Any driver who paid his fine penalty can lodge his complaint on the penalty to the Ministry of Transport for cancelation within 30 days; the decision to be given by Ministry shall be the final administrative decision.

BR-06: A traffic controller who receives driving qualification license from a driver shall notify to the driver from where he can collect his license and deliver the license to the nearby concerned body within two consecutive working day**s.**

BR-07: The fine imposed for violation of traffic Regulations shall enter in the judgment register and stay valid for one-year as if a Traffic control office believes that a traffic controller has committed offence while enforcing the provisions of this Regulation, without prejudice to his liability in accordance with appropriate law, shall remove him from his traffic controlling.

BR-08: -When they are directing traffic, authorized officials shall be easily identifiable at a distance, at night as well as by day.

2.5 Review of related systems

The following study represents a brief introduction of the papers we are using in our research project.

"Automated Over Speeding Detection and Reporting System'' [1]

The system involves a speed gun which is used to check for over-speeding of vehicles by placing it in the direction of moving vehicle. This involves manpower with a person holding the gun. If over-speeding is detected, the person informs the Toll where the vehicle can be charged.

"Design & Analysis of Vehicle Speed Control Unit Using RF Technology" [2]

The main objective is this project is to design a Smart Display controller meant for vehicle's speed control and monitoring of zones, which can run on an embedded system. Smart Display & Control (SDC) can be custom designed to fit into a vehicle's dashboard, and displays information on the vehicle. The project is comprising of two separate modules: zone status transmitter unit and receiver (speed display and control) unit. Once the information is received from the zones, the vehicle's embedded unit automatically alerts the driver, to reduce the speed according to zones, it waits for few seconds, and otherwise vehicle's SDC unit automatically reduces the speed.

“Detection of Over Speeding Vehicles on Highways” [3]

The entire implementation requires an IR transmitter, an IR receiver, a control circuit and a buzzer. The speed limit is set by the police who use the system depending upon the traffic at the certain location. The time taken by the vehicle to travel from one set point to the other is calculated by control circuit and displayed on seven segment displays. Moreover, if the vehicle crosses the speed limit, a buzzer sounds alerting the police.

CHAPTER THREE

PROPOSED SYSTEM

3.1 overview

This project has three design phases. First, we design the Over Speed Detector device(controller) for vehicle with the use of Arduino, GPS, and GSM technology. The vehicles are fitted with over speed detector device which has the capability for sharing information, recording and storing the data about the speed of vehicle if the vehicle exceeds sated speed limit, then it notifies to the central control system. The message will be displayed on LCD also.

The second part which is the central control system we are going to develop, has the ability to send information of the vehicle that has committed speed limit violation, to the nearest traffic police after receiving it from controller. Traffic police's get this information using the android app we are going to develop and take action and report to the central control system what kind of action they have taken.

3.2 Functional Requirements

3.2.1 The functionality of controller

FR-01: The controller must have detected speed of the vehicle travelling during its all journey using information from satellite.

FR-02: The controller must have to compare the speed of the vehicle it received via GPS from satellite with the sated speed limit to check whether the vehicle exceed speed limit or not.

FR-03: The controller must have to track the location of the vehicle.

FR-04: The controller must have to send an alert to the central monitoring system as soon as the vehicle current speed exceeds the sated speed limit.

FR-05: The controller must have to send the following information to central monitoring system: Plate number, speed, time, longitude and latitude values.

FR-06: The controller must have to display the vehicle over speeding event in the LCD display.

3.2.2 The functionality of central monitoring system (evsc system)

FR-07: The system must allow the admin to login to manage the entire system.

FR-08: The system must have to receive alert sent from the controller.

FR-09: The system must have to store the received alert and display on the web server.

FR-10: The system must have to find the nearest traffic police location to the vehicle and send over speeded event notification to them.

FR-11: The system must have to log received reports from traffic police with vehicle appropriate plate number.

FR-12: The system shall display the vehicle detail, traffic police detail and shall allow searching such information.

FR-13: The system shall allow adding new traffic police and registering new vehicle into the system.

3.2.3 The functionality of evsc android app

FR-14: The system must have to authenticate traffic police before enter to the system.

FR-15: The system must have to receive notification sent from the EVSC system.

FR-16: The system shall allow traffic polices to send report on the action they have taken.

FR-17: The system must have to allow traffic police to change their passwords.

3.3 Nonfunctional Requirement

3.3.1 Usability requirements

The objective is to ensure that user and usability requirements are well defined and integrated into Real-time vehicle over speed detector requirements specification. The purposes of usability methods at this stage are to collect information about the hardware devices, tasks and environments, and to agree what aspects should be formalized as requirements.

- ✓ The hardware device which will be used in vehicle should be unsophisticated and affordable.
- ✓ These hardware devices should be tiny and could be easily fixed on tracking vehicles.
- ✓ While it is in operation, the activity of such vehicle unit should not be identifiable to the person who drives the vehicle.

3.3.2 Reliability requirements

The system should be designed for maximum reliability and flexibility. Thus, the following reliability requirements were identified.

- ✓ Low-cost reliable vehicle over-speed detecting device has to be installed in the vehicles.
- ✓ The proposed system will be totally relied on the GSM network. Therefore, the reliability of those dependent services will be taken into consideration when measuring the reliability of the proposed system.

3.3.3 Performances requirements

Our system performance characteristics are outlined in this section. Include specific response times, accuracy and the system performance.

- ✓ Has to offer real-time speed detecting with minimum latency. Latency will depend on the internet speed as well as other factors. It should be noted that latency is not an issue in the context of the use of this system. Therefore, a latency of 30 seconds is deemed acceptable.
- ✓ In order to get coordinate to the system accurately, the car unit should operate properly without sending erroneous data.

- ✓ Response time should be minimal. It should be less than three minutes. 3.3.4 Hardware interface

Specific hardware requirements for vehicle unit and monitoring station machine that, vehicle overspeed detecting device going to be resided on, have been affirmed as follows.

- ✓ The software system should be running on a computer workstation or server class PC or a high-performance laptop PC with minimum of following configuration. CPU: Pentium-IV or newest/ RAM: 512Mb (minimum) / HDD: 30 GB / VGA Mem: 64Mb / 56K Modem / (IR Optional).
- ✓ Vehicle overspeed detecting device should be a portable one and should be battery powered & rechargeable.
- ✓ Vehicle overspeed detecting device should be mounted and hidden to the user (i.e.: Driver).

3.4 system Model

3.4.1 Scenario

Scenario 1: Login

Actor: Traffic police

Initial assumption: Traffic polices have the application “application name” installed on their mobile phone.

Normal: when clicked on for the first time, application displays the login page to the Traffic police. Then they enter their respective username and password to the login form they are directed to. After filling the form, they click on the login button and if they have entered correct password and username, they will be redirected to homepage.

What can go wrong: In case they have entered wrong username and password, or correct username but wrong password, system prompts error message to enter the password and the username again and redirects them to the login page.

System state on completion: System will save the information to the database and traffic police will be allowed to view the homepage.

Scenario 2: Traffic police registration

Actor: Admin

Initial assumption: Admin have logged into the system and he/she is on the homepage of the systems web address.

Normal: Admin clicks on “Register” from menu and from the lists select ‘Traffic police’ and fill in the form displayed with appropriate traffic police information and after finishing it clicks on register.

What can go wrong: By mistake, if the admin clicks on ‘register’ before filling all the required fields, then the system will remind to fill all the required fields.

System state on completion: Filled in information of the traffic police being added will be saved to database.

Scenario 3: Vehicle registration

Actor: Admin

Initial assumption: Admin have logged into the system and he/she is on the homepage of the systems web address.

Normal: Admin clicks on “Manage” from menu and from the lists select ‘Vehicle’ and again from the dropdown selects ‘add’. Then fills in the form displayed with appropriate vehicle information and after finishing it clicks on register.

What can go wrong: By mistake, if the admin clicks on ‘register’ before filling all the required fields, then the system will remind to fill all the required fields.

System state on completion: Filled in information of that particular Vehicle being added will be added will be saved to database.

Scenario 4: Change Vehicle State

Actor: Admin

Initial assumption: Admin have logged into the system and he/she is on the homepage of the systems web address.

Normal: Admin clicks on “Manage” from menu and from the lists select ‘Vehicle’ and again from the dropdown selects ‘change state’. Then search for the vehicle information needed to be change its state by its plate number, using the search bar. When the result is returned, click on the ‘state’ button available in front of it and change state either to active or passive.

What can go wrong: Admin may try to remove unregistered vehicle from the system. If this happens, then display error message.

System state on completion: Vehicle state will be updated.

Scenario 5: change Traffic Police state

Actor: Admin

Initial assumption: Admin have logged into the system and he/she is on the homepage of the systems web address.

Normal: Admin clicks on “Manage” from menu and from the lists select ‘Traffic police’ and again from the dropdown selects ‘change state’. Then search for the specific traffic police using the search bar. When the result is returned, click on the ‘state’ button available in front of it and change state either to active or passive.

System state on completion: Traffic police state will be updated.

Scenario 6: Send Report

Actor: Traffic Police

Initial assumption: A reporting traffic police have logged into the mobile application and is on the homepage.

Normal scenario: Reporting traffic police clicks on ‘View Notification’ and from that page clicks on the three dots found in front of the selected notification. Then fills in required information’s of the vehicle and finally clicks on ‘send’ button after finishing.

Other Scenario: Traffic police clicks on Send report and from the lists displayed selects what kind of action is taken. Fills in the required information accordingly and clicks on ‘Send’. In case the Vehicle has not responded to him/her, that traffic police will click on ‘Send report’ from homepage and fill in the form displayed. And click on ‘Send’ after finishing.

What can go wrong: By mistake, if the reporting traffic police clicks on send before filling all the required fields, then the system will remind to fill all the required fields.

System state on completion: Action taken by traffic police will be will be sent out to the central control system where the admin is managing and be stored int the database.

Scenario 7: View Notification

Actor: Traffic police

Initial Assumption: Traffic police have logged into the mobile application and is on the homepage.

Normal: Traffic police clicks on ‘view notification’ button from the menu and clicks on notification sent from central control system to view the detail of it.

What can go wrong: Maybe incomplete notification is sent to the traffic police. Send a request to resend the data again.

System state on completion: The system will store that the sent notification was viewed by the traffic police.

Scenario 8: Track location and speed

Actor: Controller

Initial Assumption: The controller is attached to the vehicle and is functioning properly.

Normal: GPS found in the controller will look up on the google map and track the location of the vehicle in the longitude and latitude refreshing itself by five minutes’ interval.

What can go wrong: If the controller is not functioning properly, send a report to the central control system so that they get it fixed.

System state on completion: Vehicle’s location and speed database will be updated.

Scenario 9: Send alarm

Actor: Controller

Initial Assumption: The controller is attached to the vehicle and is functioning properly.

Normal: Controller compares the current speed of the vehicle with the stated speed limit for that particular road and if the Vehicle goes with speed above the sated one, then an alarm will be triggered and sent to the central control system automatically through the GSM found in the controller.

What can go wrong: If the controller is not functioning properly, send a report to the central control system so that they get it fixed.

System state on completion: Sent alarm will be saved to the database.

Scenario 10: Display

Actor: Controller

Initial Assumption: The controller is attached to the vehicle and it is functioning properly.

Normal: Controller displays on the LCD the current speed by which the vehicle is travelling, current location, and the maximum speed allowed for that road. In case over-speeding happened, it will display it to the driver that it has surpassed the maximum speed limit allowed.

What can go wrong: If the controller is not functioning properly, send a report to the central control system so that they get it fixed.

System state on completion: Vehicle's location and speed database will be updated.

Scenario 11: Receive alarm

Actor: EVSC system

Preconditions: EVSC system is up and running

Normal: when detecting the speed with which the vehicle is going is above the sated limit, the controller will send out an alarm to the EVSC system and it will receive it.

What can go wrong: After the alarm is sent from the controller, due to network error alarm maybe delayed. Solution will be re-sending that alarm.

Scenario 12: Find nearest Traffic Police

Actor: EVSC system

Preconditions: EVSC system is up and running.

Normal: Up on receiving an alarm from the controller, EVSC system will find out the nearest traffic Police to the place where the vehicle has committed over-speed, using an algorithm.

What can go wrong: Traffic police may not be available at the position he/she have been assigned.

Scenario 13: Receive Report

Actor: EVSC system

Precondition: Traffic police have sent a report after taking action on the vehicle that has over-speeded.

Normal: Traffic police sends a report when he/she have taken an action and EVSC system receives it.

What can go wrong:

System state on completion: sent report data is added to the database.

Scenario 14: View report

Actor: Admin

Precondition: EVSC system has received report from traffic police via the EVSC mobile app and admin is on the homepage of the system.

Normal: Admin goes to view report and clicks on it.

What can go wrong:

System state on completion: Database state will not be affected, since it is view only done here.

Scenario 15: View alarm

Actor: Admin

Precondition: The controller attached to the vehicle has sent an alarm and admin is on the homepage of the system.

Normal: Admin from homepages goes to the view menu and clicks on it, then from the displayed dropdown menus, selects alarm and clicks on it.

Scenario 16: View Vehicle detail

Actor: Admin

Precondition: Admin is on the homepage of the system.

Normal: Admin from homepages goes to the view menu and clicks on it, then from the displayed drop-down menus, selects Vehicle detail and clicks on it.

System state on completion: Database state will not be affected, since it is view only done here.

Scenario 16: View Traffic Police profile

Preconditions: Admin is on homepage of the system.

Normal: Admin from homepages goes to the view menu and clicks on it, then from the displayed drop-down menus, selects traffic police profile and clicks on it.

What can go wrong: If that traffic police data is not found in the database, prompt to Admin that there is no data for that particular search.

System state on completion: Database state will not be affected, since it is view only done here.

Scenario 17: View Vehicle over-speed history

Actor: Admin

Preconditions: Admin is on homepage of the system.

Normal: Admin from homepages goes to the view menu and clicks on it, then from the displayed drop-down menus, selects Vehicle over-speed history and clicks on it.

What can go wrong: If there is no data in the database registered by that particular search of the vehicle plate, prompt to the admin to refine its search.

System state on completion: Database state will not be affected, since it is view only done here.

What can go wrong: while filling in the forms, if he/she might have forgotten to fill in some information's or may have filled in wrongly, then show an error message and prompt to correct it.

System state on completion: Sent report data will be stored in database.

Scenario 18: Detect speed

Actor: Controller

Precondition: controller is attached to the vehicle and is working properly.

Normal: Based on the given road speed limit the controller will detect the speed with which the vehicle is moving from the satellite via GPS.

3.4.2 Use case model

Actor Identification

The actors of our system are:

- ✓ Controller
- ✓ Traffic Police
- ✓ Admin
- ✓ Super Admin
- ✓ EVSC System

Use case Diagram

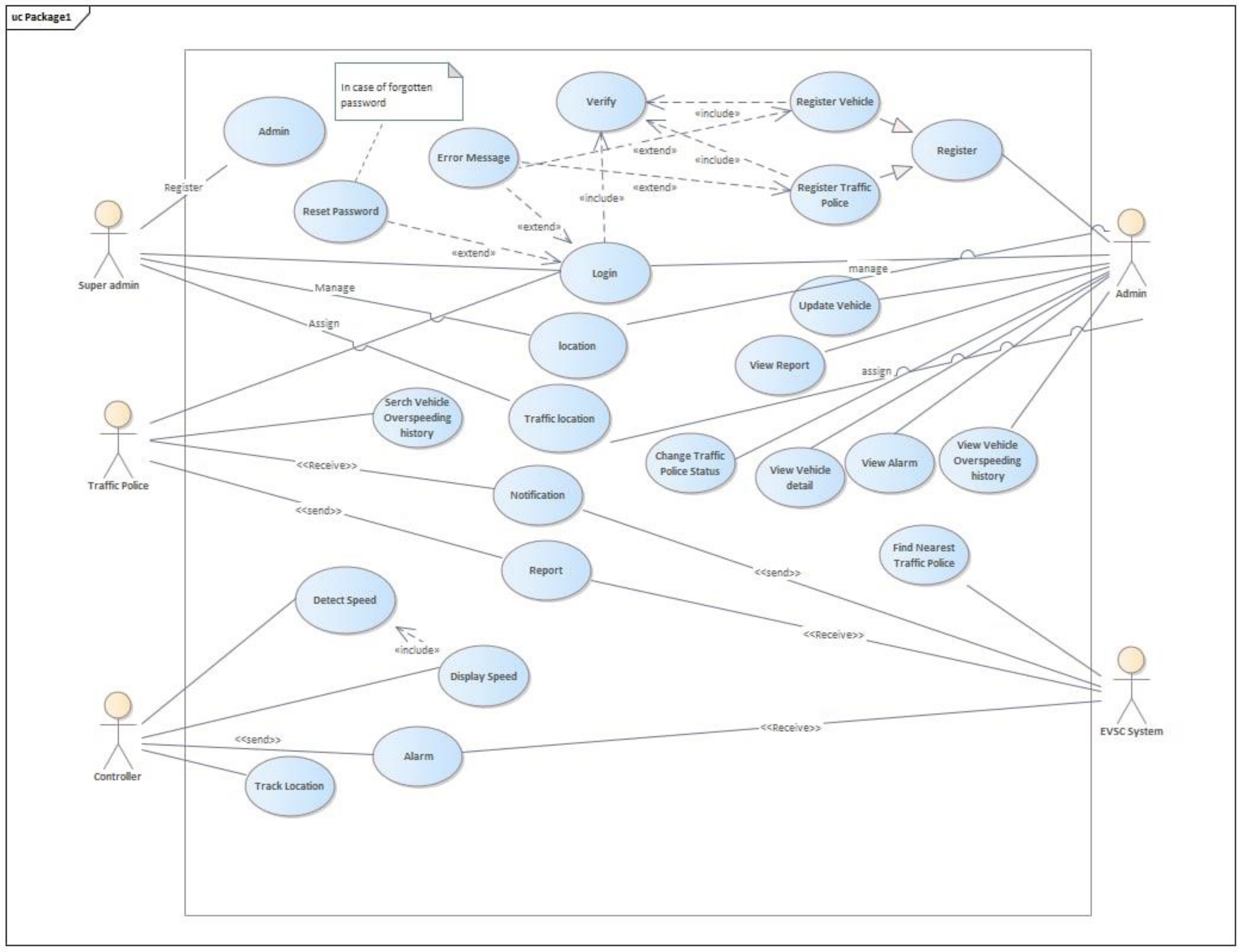


Figure 3. 1 use case diagram

3.4.3 Use case description

Use case name	Login
Use case number	Uc01
Description	A Traffic Police Login to the System to access System functionality
Participating actor	Traffic Police
Pre-condition	Must have Application and Account Must connect to the network
Flow of event	➢ The Traffic Police must have Application ➢ The Traffic Police opens The Application ➢ System Display Home Page with login ➢ Traffic Police enters username and Password to login ➢ If the password and username are correct system opens home page
Post condition	After successful login Home Page is available
Alternative flow of event	If Traffic Police enters wrong username and password system prompts error message and redirect to login page (redirect to step 3).

Table 3. 1 use case description of login

Use case name	Receive Alarm
Use case number	Uc02
Description	EVSC System can receive Alarm Triggered by Controller that is attached on Vehicle

Participating actor	EVSC System
Pre-condition	✓ Controller on the vehicle must send the required data ✓ Network must be available
Flow of event	1. GPS module track location and calculate speed 2. GSM module Trigger Alarm from vehicle to the EVSC System 3. Triggered Alarm will be stored on the database
Post condition	Traffic police and Admin can able to access the data
Alternative flow of event	

Table 3. 2 use case description of Receive alarm

Use case name	Send Report
Use case number	Uc03
Description	Traffic police sends the report to the central system about the action taken in response to data from vehicle
Participating actor	Traffic Police
Pre-condition	• Access to vehicle location and over speeding information • Network must be available
Flow of event	1. GPS module track location and calculate speed

	2. GSM module send data from vehicle to the central database 3. Data will be stored on the database 4. Android App will be notified with new data and the data can also be accessed from central website. 5. Traffic police use the data and take some action. 6. Traffic Police sends report to central system and Report will be stored on central database.
Post condition	Central System administrator can view the report
Alternative flow of event	If the track refused to

Table 3. 3 use case description of Send report

Use case name	Track Location
Use case number	Uc04
Description	Controller Track location of vehicle with the help of GPS Module.
Participating actor	Controller
Pre-condition	➢ Device must have power to work
Flow of event	➢ GPS receivers use a constellation of satellite and ground station to compute position.

Post condition	➢ Location will be accessible to GSM module for later use.
Alternative flow of event	

Table 3. 4 use case description of Track location

Use case name	Detect speed
Use case number	Uc05
Description	Controller calculate speed of vehicle and compare to default permitted speed value.
Participating actor	Controller
Pre-condition	➢ Device must have power to work
Flow of event	➢ GPS receivers use a constellation of satellites and ground stations to compute position. ➢ Then the speed can be calculated from the location and time it takes.
Post condition	➢ Detected speed will be accessible to GSM module for later use.
Alternative flow of event	

Table 3. 5 use case description of Detect speed

Use case name	Send Alarm
Use case number	Uc06

Description	Controller Send Alarm to the EVSC System when vehicle commit over speeding.
Participating actor	Controller
Pre-condition	➢ Controller must know the Location as well as Speed value
Flow of event	➢ Controller must compare speed of vehicle to previously settled limit. ➢ If the speed of vehicle is above the limit the Alarm will be sent to EVSC System.
Post condition	Data will be stored on the database
Alternative flow of event	

Table 3. 6 use case description of Send Alarm

Use case name	Send Notification
Use case number	Uc07
Description	EVSC System notifies nearest Traffic police Specific vehicle that has committed over speeding.
Participating actor	EVSC System
Pre-condition	➢ EVSC System must receive alarm from Controller. ➢ Traffic Police must have access to network.
Flow of event	➢ After accepting Alarm from Controller ➢ EVSC System notifies The Nearest Traffic Police

Post condition	Traffic Police access that Notification

Table 3. 7 use case description of send notification

Use case name	Display speed
Use case number	Uc08
Description	LCD displays used to display speed status to the front of vehicle.
Participating actor	Controller
Pre-condition	Data required from GPS module
Flow of event	✓ Access data from GPS module using Arduino. ✓ Display the data.
Post condition	Data will be displayed on LCD display.
Alternative flow of event	

Table 3. 8 use case description of display speed

Use case name	Receive Report
Use case number	Uc09
Description	EVSC System can receive the report from Traffic Police
Participating actor	EVSC System

Pre-condition	Report must be initiated from Traffic Police
Flow of event	✓ Report from Traffic police is arrived ✓ Report will be stored.
Post condition	Report will be available to Admin.
Alternative flow of event	

Table 3. 9 use case description of receive report

Use case name	View Report
Use case number	Uc10
Description	Admin can view Report from Traffic Police
Participating actor	Admin
Pre-condition	Report must be available on the database
Flow of event	✓ Admin Navigate through the Page to find a link which is dedicated to show page that contain report from Traffic Police.

Table 3. 10 use case description of View Report

Use case name	**Register Traffic Police**
Use case number	Uc11
Description	Admin can register Traffic Police.
Participating actor	Admin
Pre-condition	Available physically to be registered
Flow of event	✓ Admin fill the required information and hit Submit button ✓ The form will be verified.

	✓ Data will be permanently stored in the database.
Post condition	Can login and use the system using their credential.
Alternative flow of event	

Table 3. 11 use case description of Register Traffic Police

3.5 Object Model

3.5.1 Class diagram

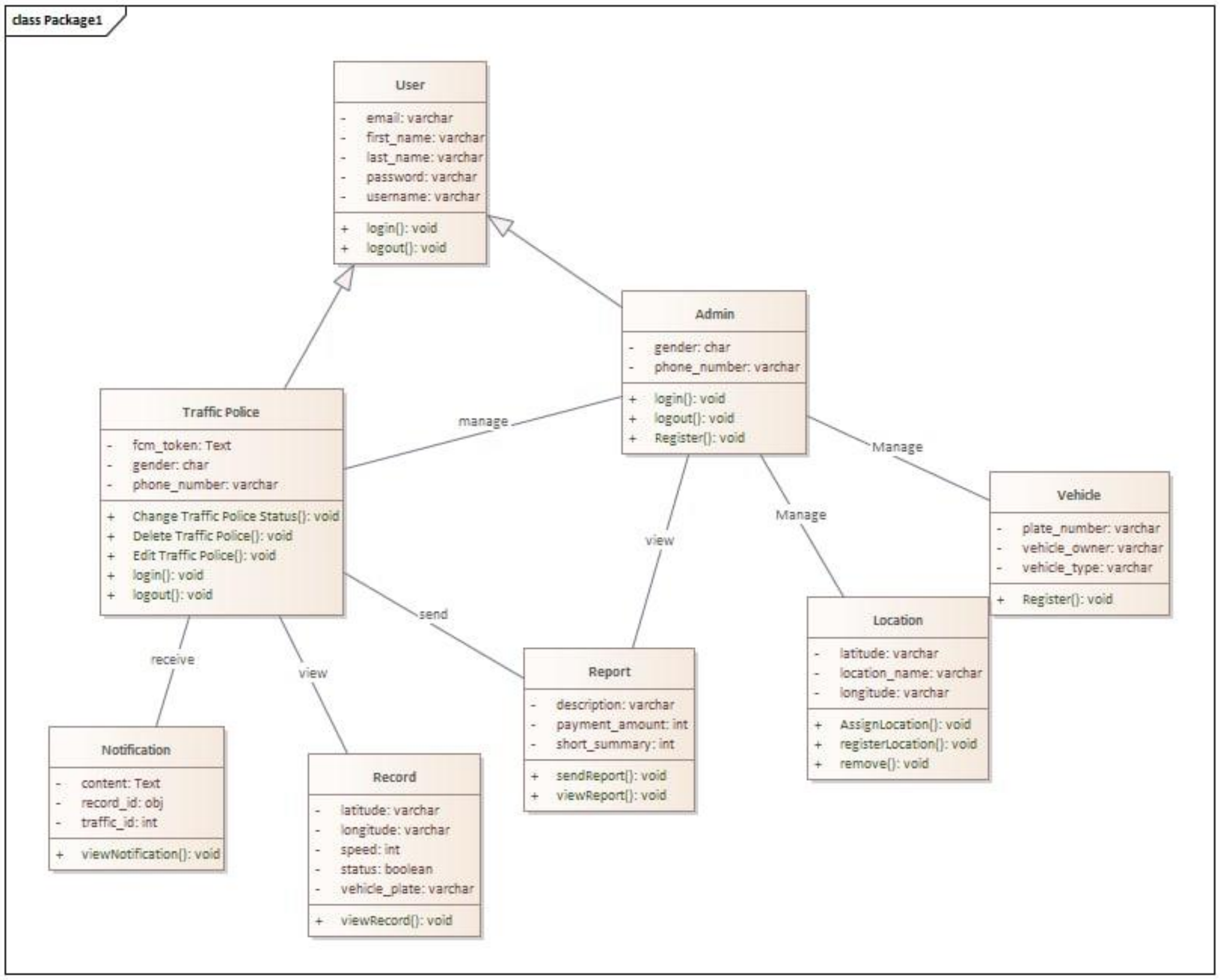


Figure 3. 2 class diagram

3.5.2 Data dictionary

Data Item	Data Type	Data format	Number of Character	Description	Example	Validation
ID	Integer		7-10	Is a unique identifier for Traffic Polices and for Admins	1	Must be unique for a table and of a proper format.
First Name	String		32	First Name for Account Holders (i.e. Traffic Polices and System Admins)	Amanuel	
Last Name	String		32	Last Name for Account Holders	Girma	
Email	String	Standard email address format	52	Email address of the Account Holder (i.e., Traffic Police and System Admins)	amanuel070@gmail.com	
Longitude	Double		46	Stores Traffic Polices' or Vehicles' longitude value of location.	A way to reference location in a map	Must be a valid coordinate value in a Map.
Latitude	Double		46	Stores Traffic Polices' or Vehicles' latitude value of location.		Must be a valid coordinate value in a Map.
Username	String		50	A unique name given to an account holder. This are either Traffic Police or EVSC system Admins.		Must be unique and of a proper format.
Password	String	******	50	Passwords are keys given or created by an		Must be unique and combination

				account holder to provide access control for the system.		of letters, special characters and numbers.
Vehicle Plate	Integer		11	Plate Number of a registered Vehicle		Must be a valid vehicle plate number format.
Vehicle Owner	String		50	Name of Vehicle Owner		
Vehicle Status	Integer		1	Vehicle status holder. (i.e. describes whether a vehicle is in a service of out of a service)	0	
Vehicle Speed	Integer		11	The Speed of vehicle recorded during over speeding.	140km/hr	

Table 3. 12 Data dictionary

3.5.3 Sequences diagram

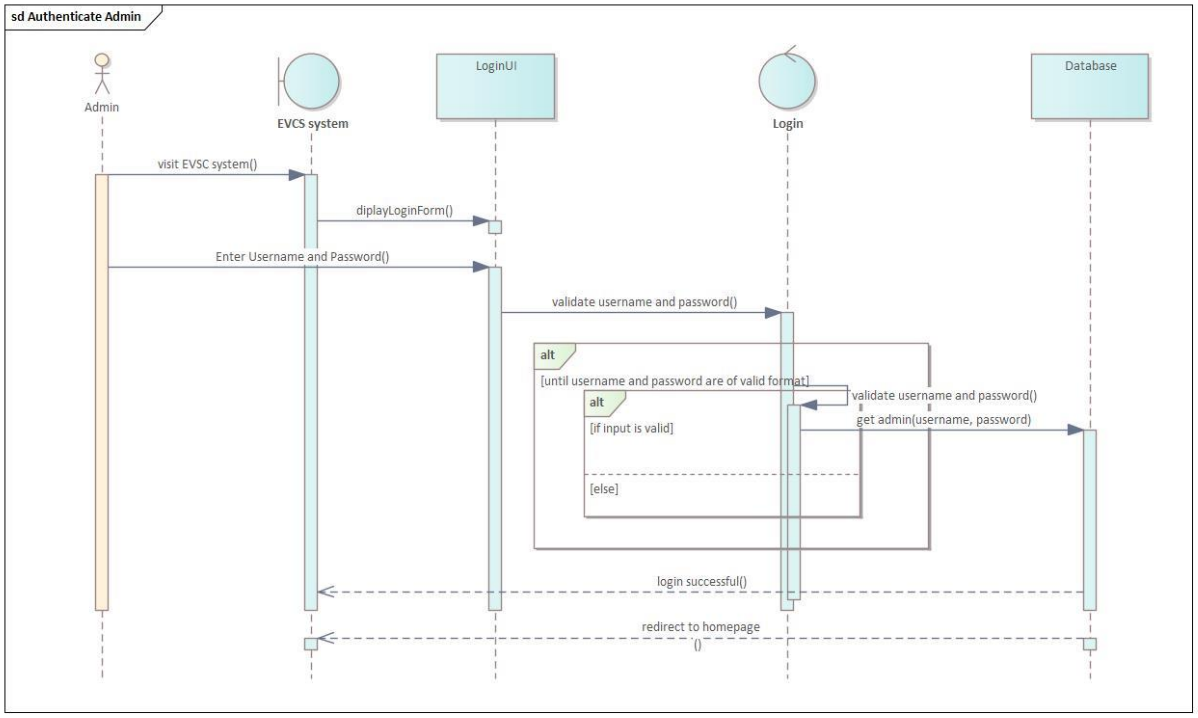

sd Authenticate Admin
Admin
EVCS system
LoginUI
Login
Database
visit EVSC system()
diplayLoginForm()
Enter Username and Password()
validate username and password()
alt
[until username and password are of valid format]
alt
[if input is valid]
validate username and password()
get admin(username, password)
[else]
login successful()
redirect to homepage
()

Figure 3. 3 sequence diagram for authenticating admin

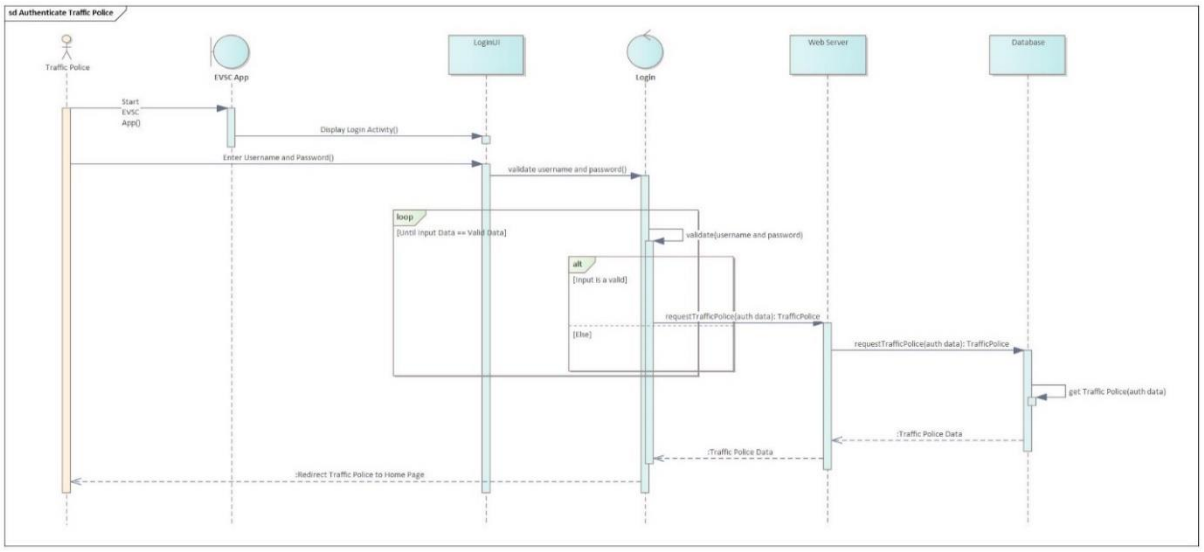

sd Authenticate Traffic Police
Traffic Police
EVSC App
LoginUI
Login
Web Server
Database
Start EVSC App()
Display Login Activity()
Enter Username and Password()
validate username and password()
loop
[Until Input Data == Valid Data]
validate(username and password)
alt
[Input is a valid]
requestTrafficPolice(auth data): TrafficPolice
[Else]
requestTrafficPolice(auth data): TrafficPolice
get Traffic Police(auth data)
:Traffic Police Data
:Traffic Police Data
:Redirect Traffic Police to Home Page

Figure 3. 4 sequence diagram for authenticating traffic police

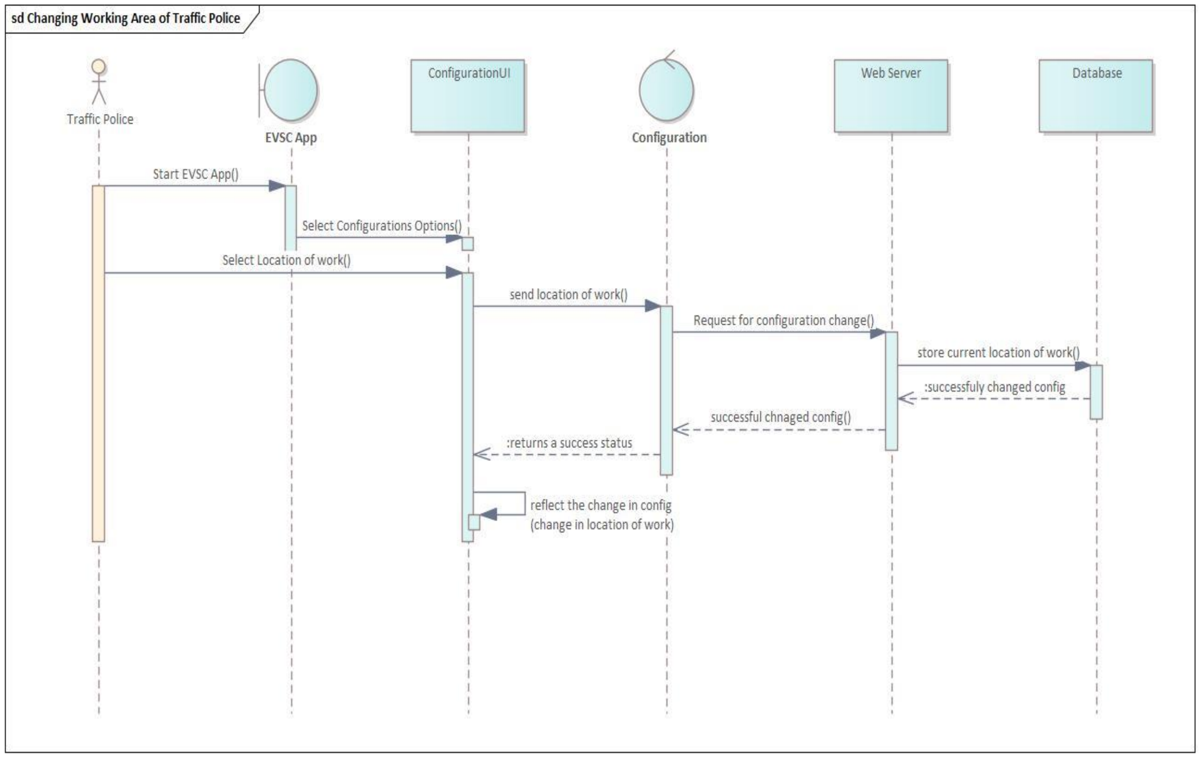

sd Changing Working Area of Traffic Police
Traffic Police
EVSC App
ConfigurationUI
Configuration
Web Server
Database
Start EVSC App()
Select Configurations Options()
Select Location of work()
send location of work()
Request for configuration change()
store current location of work()
:successfuly changed config
successful chnaged config()
:returns a success status
reflect the change in config
(change in location of work)

Figure 3. 5 sequence diagram for changing work location of traffic police

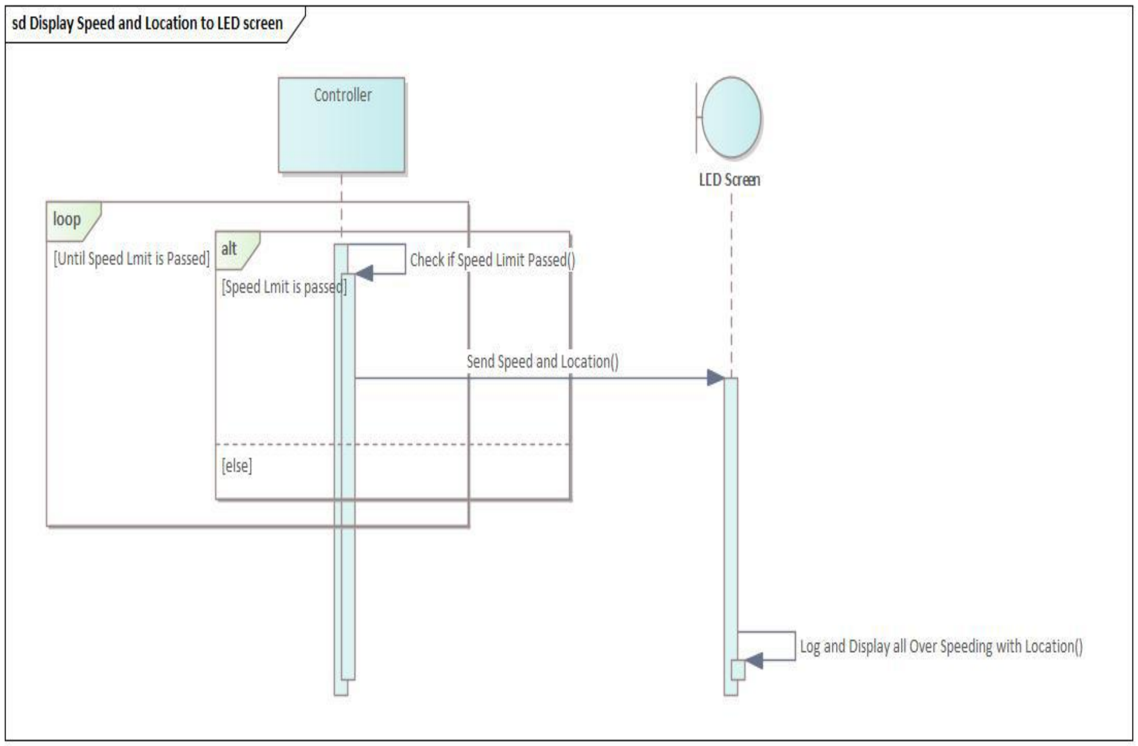

sd Display Speed and Location to LED screen
Controller
LCD Screen
loop
[Until Speed Lmit is Passed]
alt
[Speed Lmit is passed]
Check if Speed Limit Passed()
Send Speed and Location()
[else]
Log and Display all Over Speeding with Location()

Figure 3. 6 sequence diagram of displaying speed and location to LCD screen

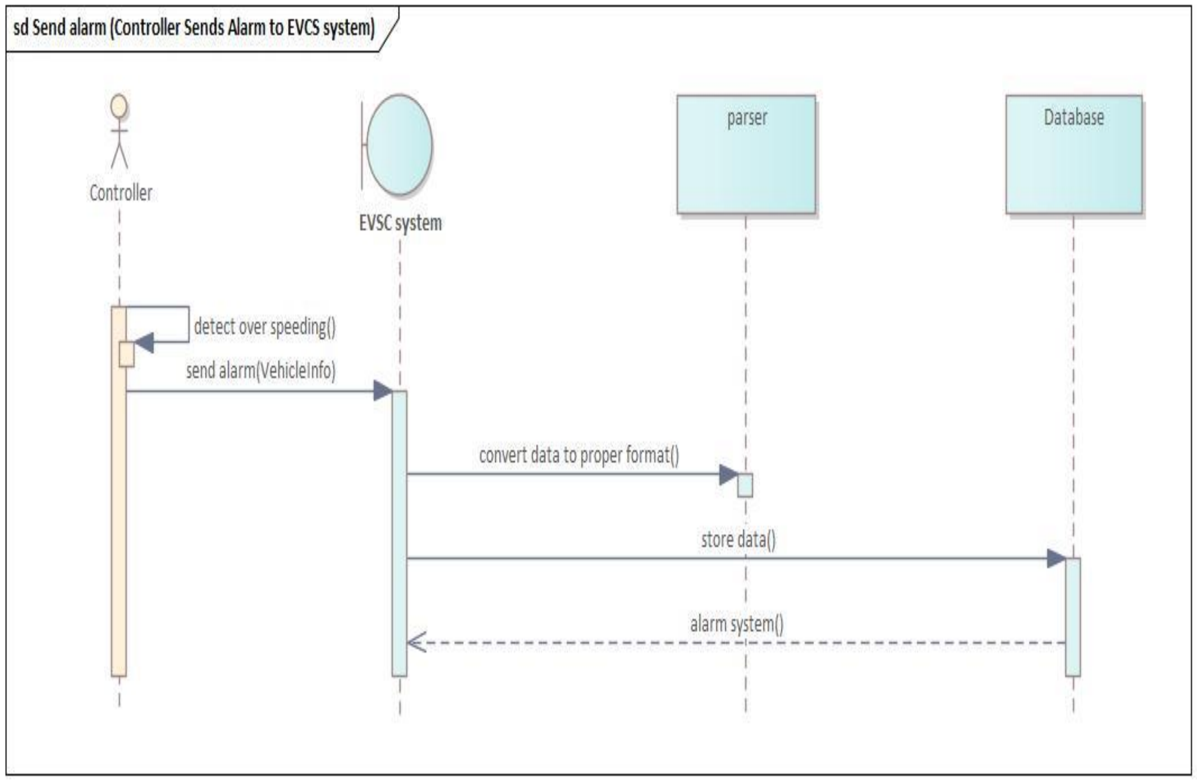


Figure 3. 7 sequence diagram for sending alarm to evsc system

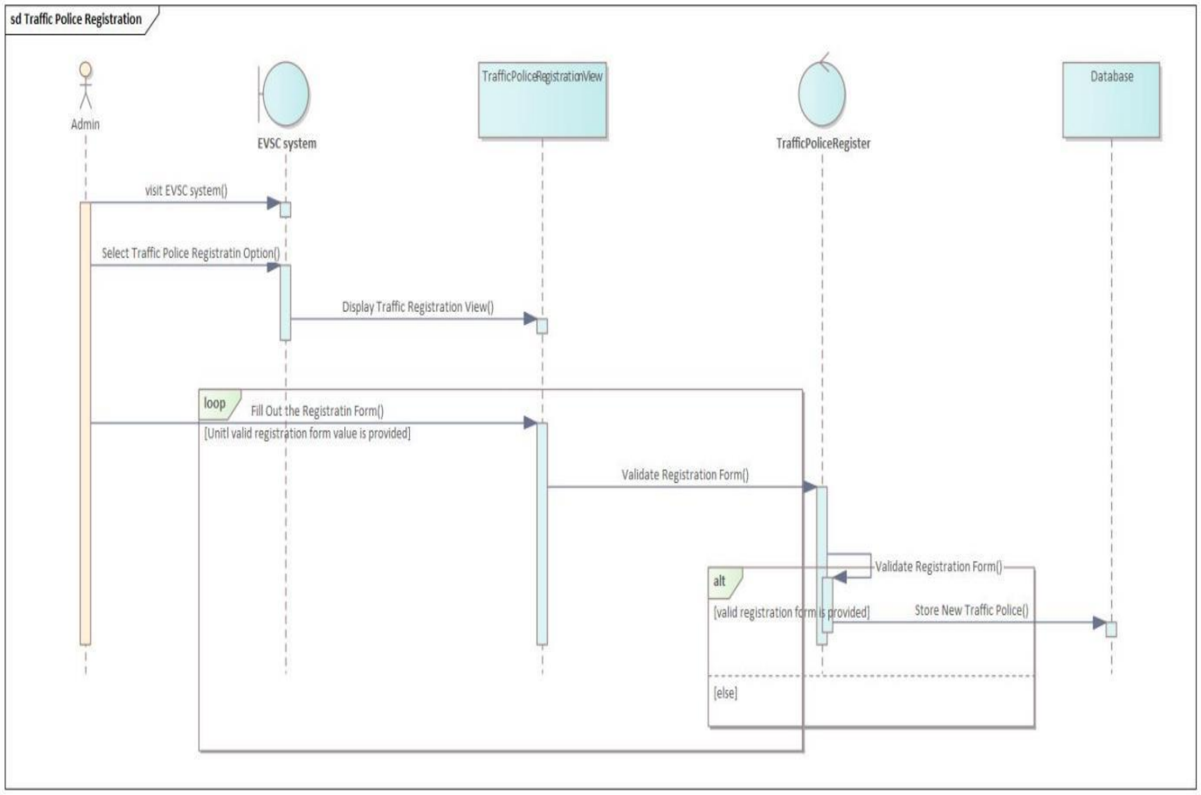

sd Traffic Police Registration
Admin
EVSC system
TrafficPoliceRegistrationView
TrafficPoliceRegister
Database
visit EVSC system()
Select Traffic Police Registratin Option()
Display Traffic Registration View()
loop
Fill Out the Registratin Form()
[Unitl valid registration form value is provided]
Validate Registration Form()
alt
Validate Registration Form()
[valid registration form is provided]
Store New Traffic Police()
[else]

Figure 3. 8 sequence diagram for traffic police registration

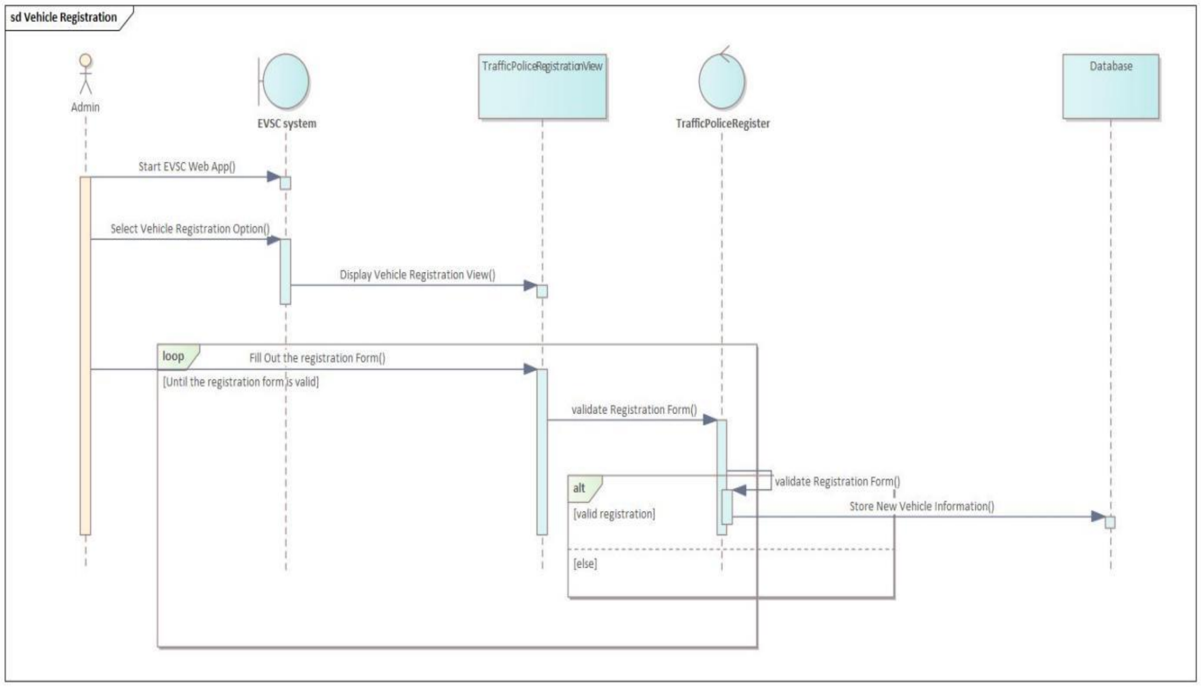

sd Vehicle Registration
Admin
EVSC system
TrafficPoliceRegistrationView
TrafficPoliceRegister
Database
Start EVSC Web App()
Select Vehicle Registration Option()
Display Vehicle Registration View()
loop
Fill Out the registration Form()
[Until the registration form is valid]
validate Registration Form()
alt
validate Registration Form()
[valid registration]
Store New Vehicle Information()
[else]

Figure 3. 9 sequence diagram for vehicle registration

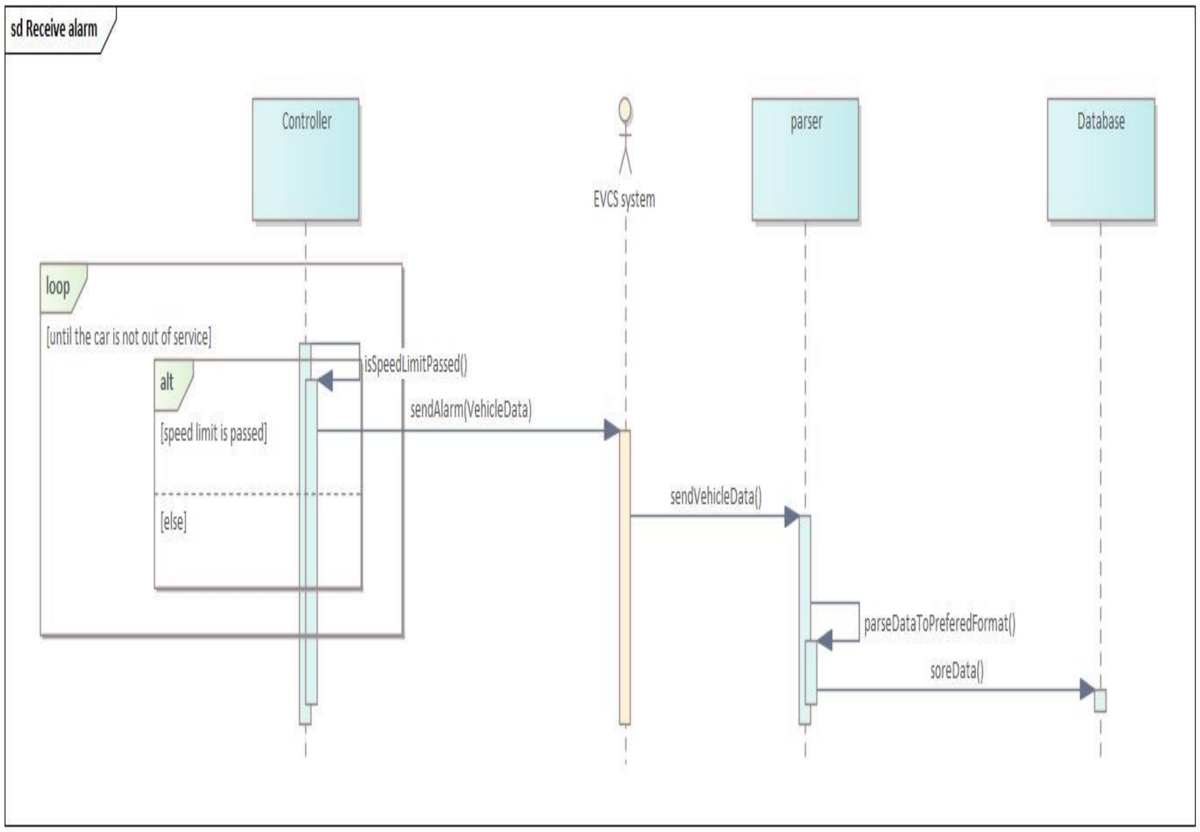


Figure 3. 10 Sequence diagram of receive alarm

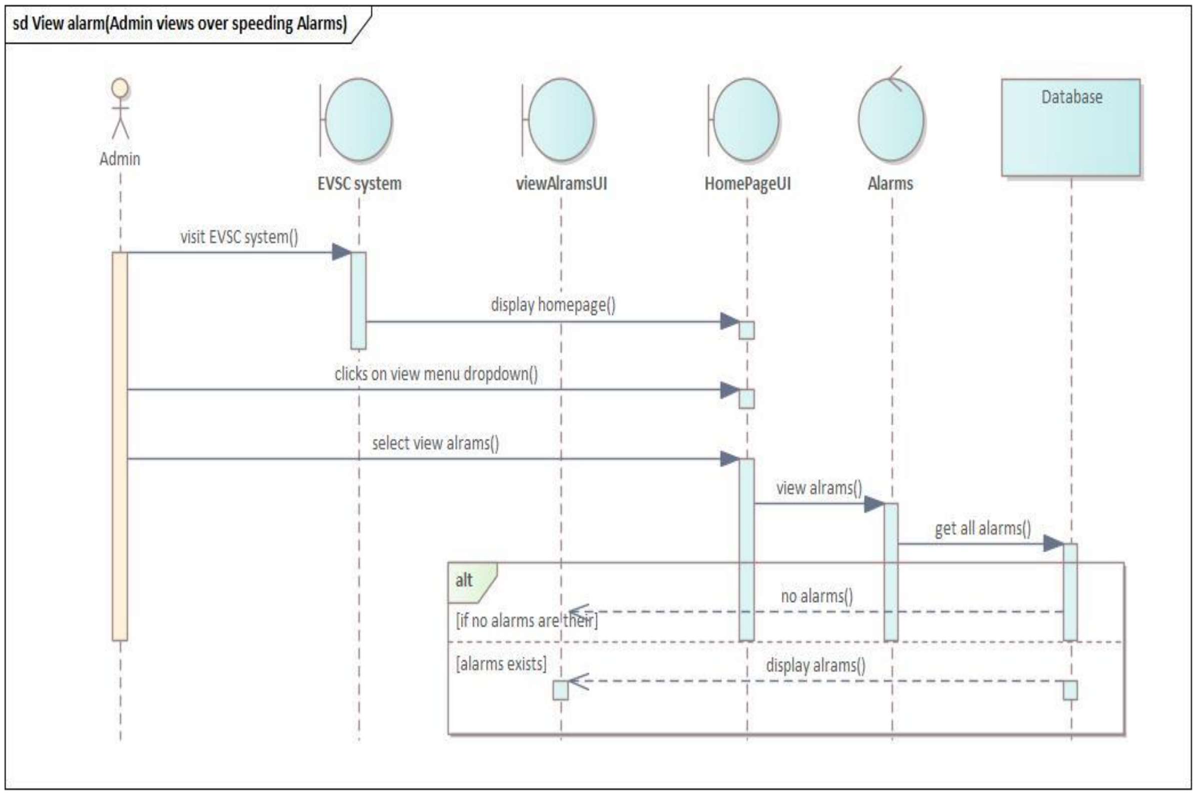


Figure 3. 11 Sequence diagram of view alarm

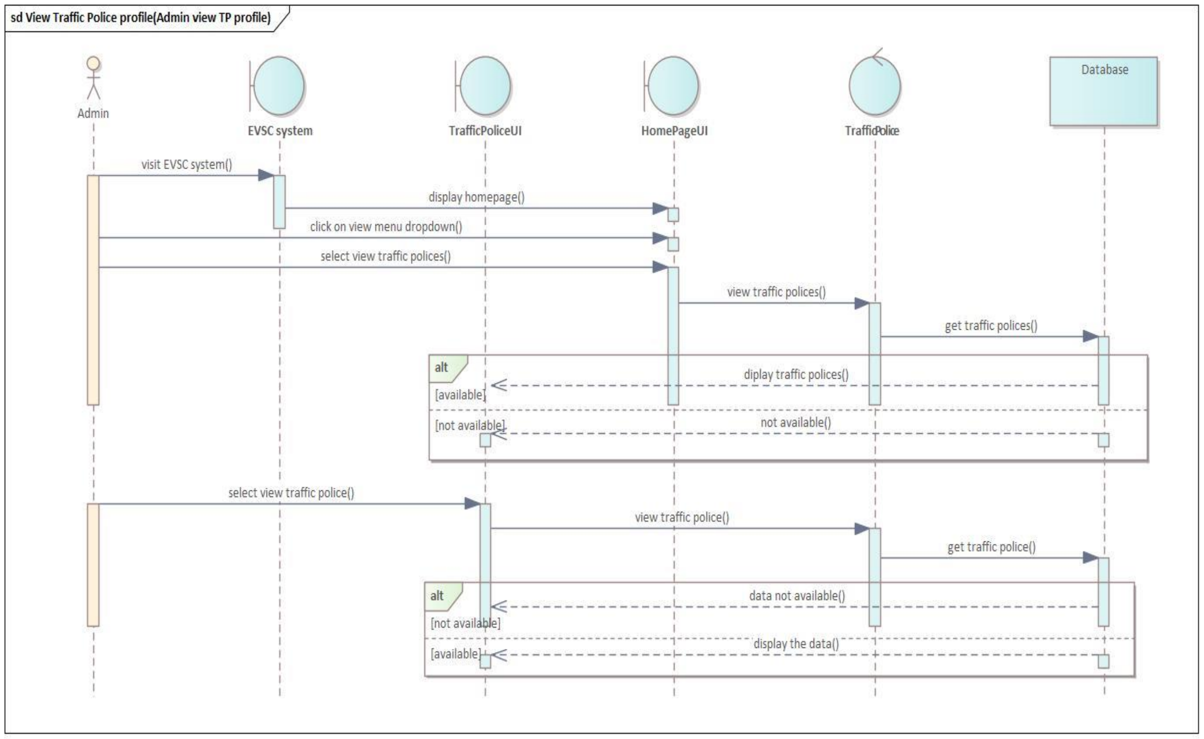


Figure 3. 12 Sequence diagram of view traffic police profile

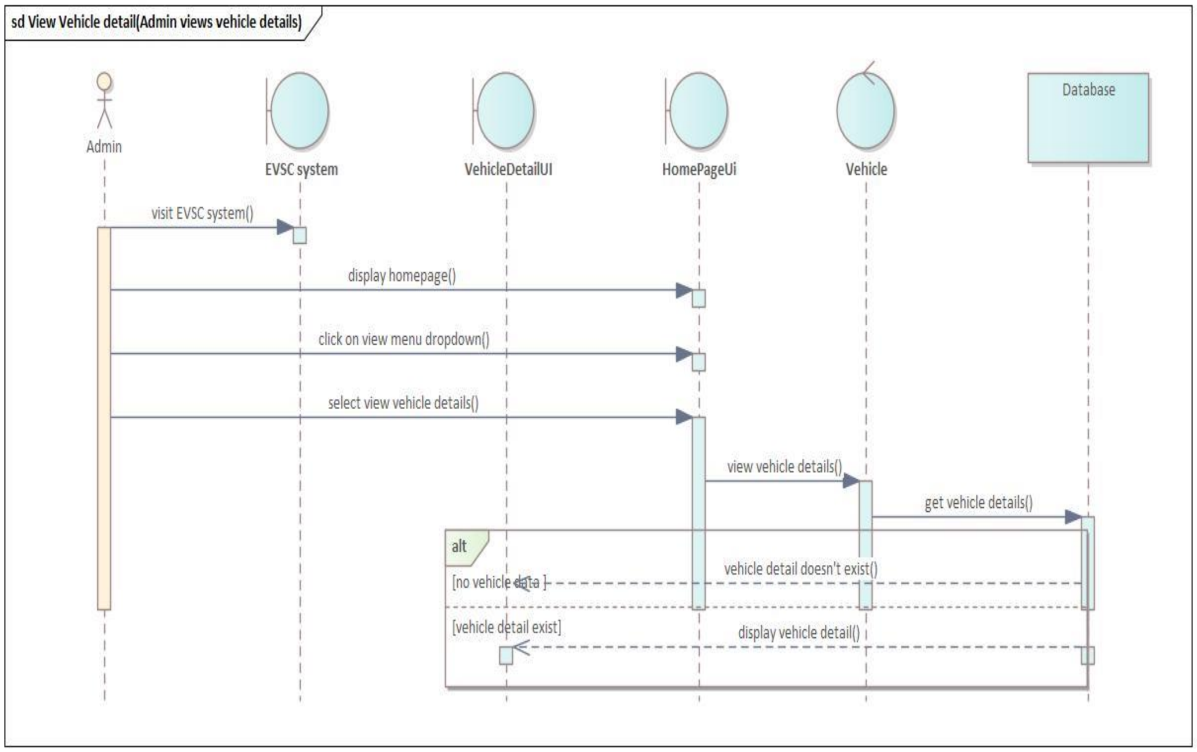


Figure 3. 13 Sequence diagram of view vehicle detail

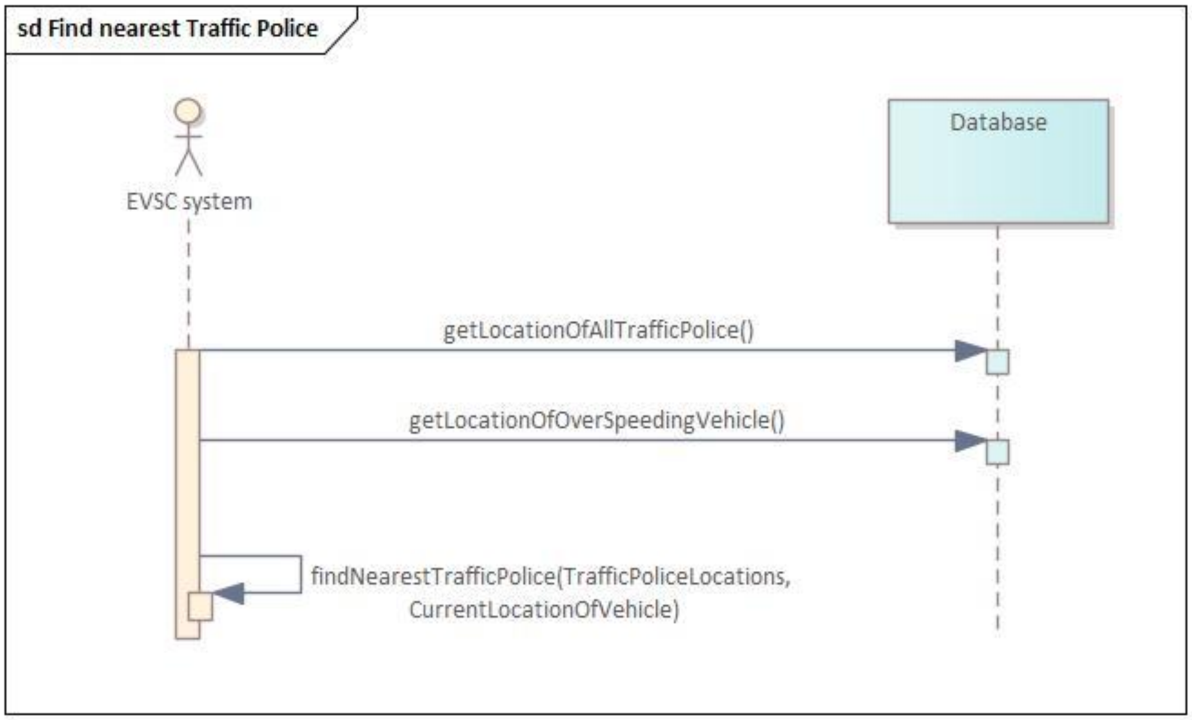

sd Find nearest Traffic Police
EVSC system
Database
getLocationOfAllTrafficPolice()
getLocationOfOverSpeedingVehicle()
findNearestTrafficPolice(TrafficPoliceLocations, CurrentLocationOfVehicle)

Figure 3. 14 Sequence diagram of finding nearest traffic police

3.5.4 Activity Diagram

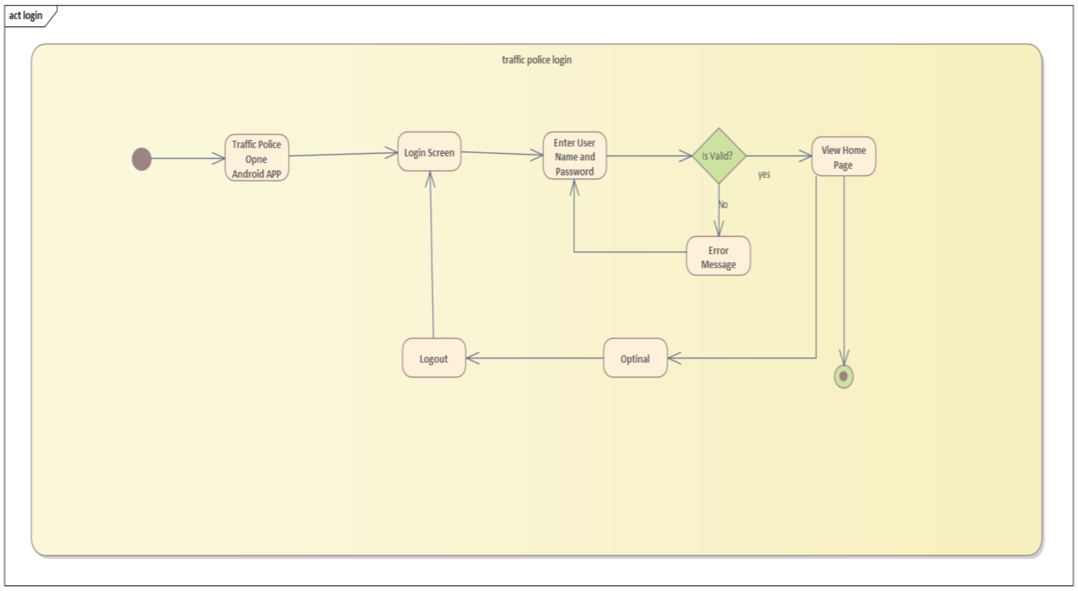


Figure 3. 15 Activity Diagram of Traffic police login to app

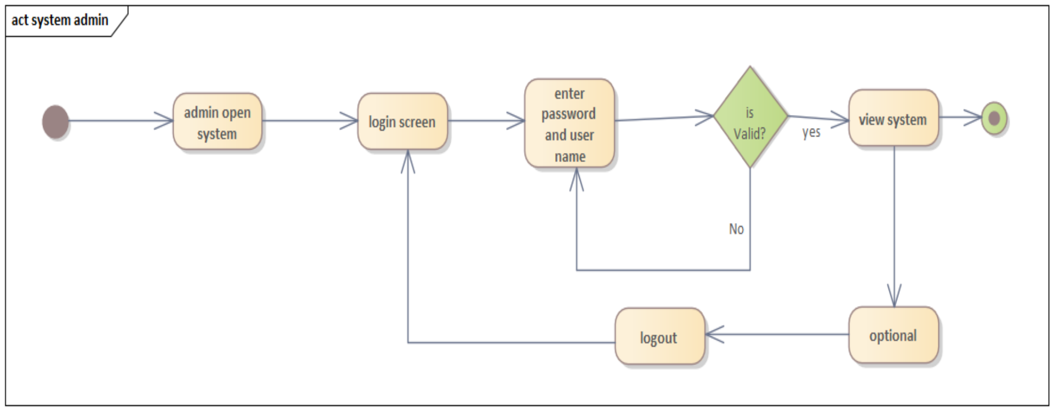


Figure 3. 16 Activity Diagram of System admin login to System

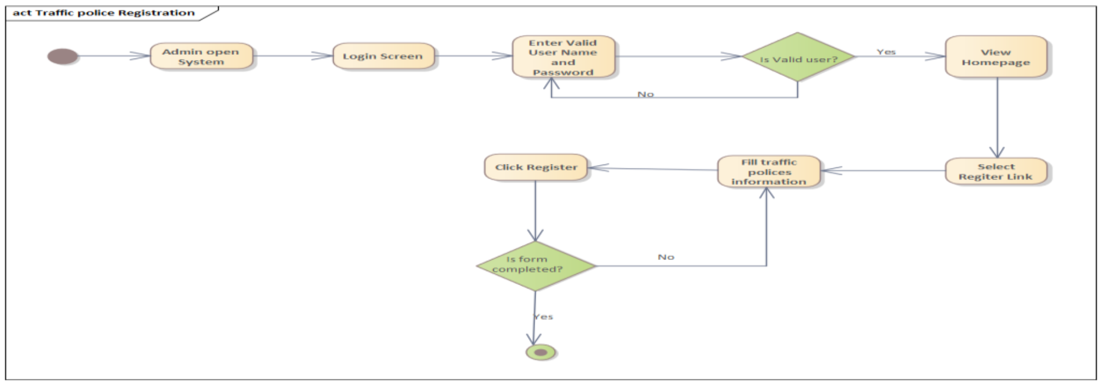


Figure 3. 17 Activity Diagram of Traffic Polices Registration

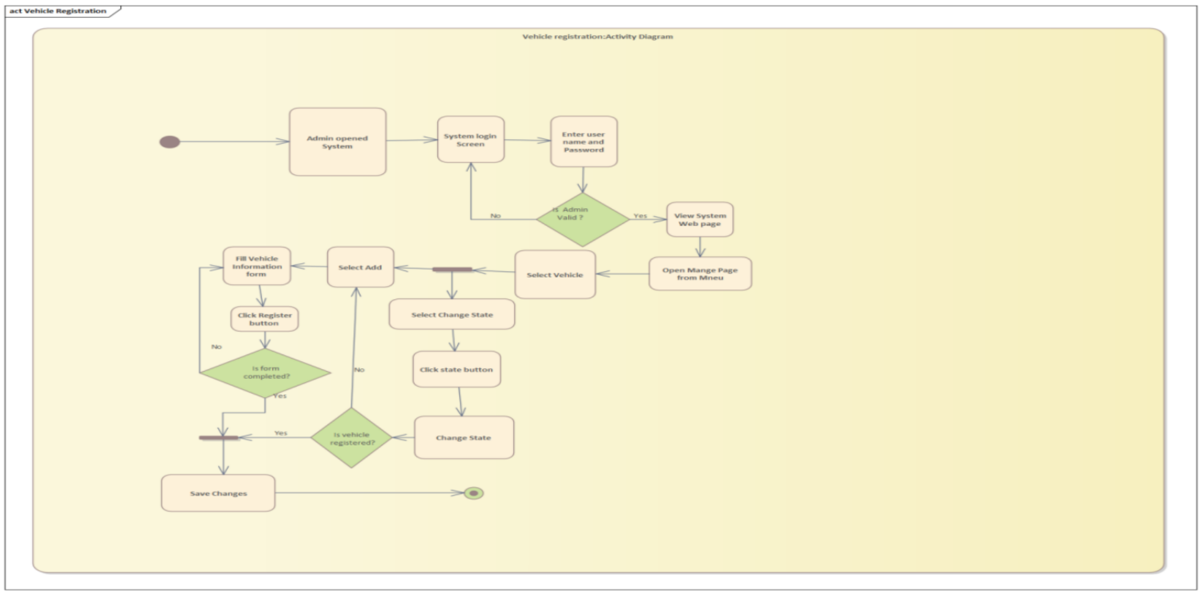


Figure 3. 18 Vehicle Registration and Change Vehicle state

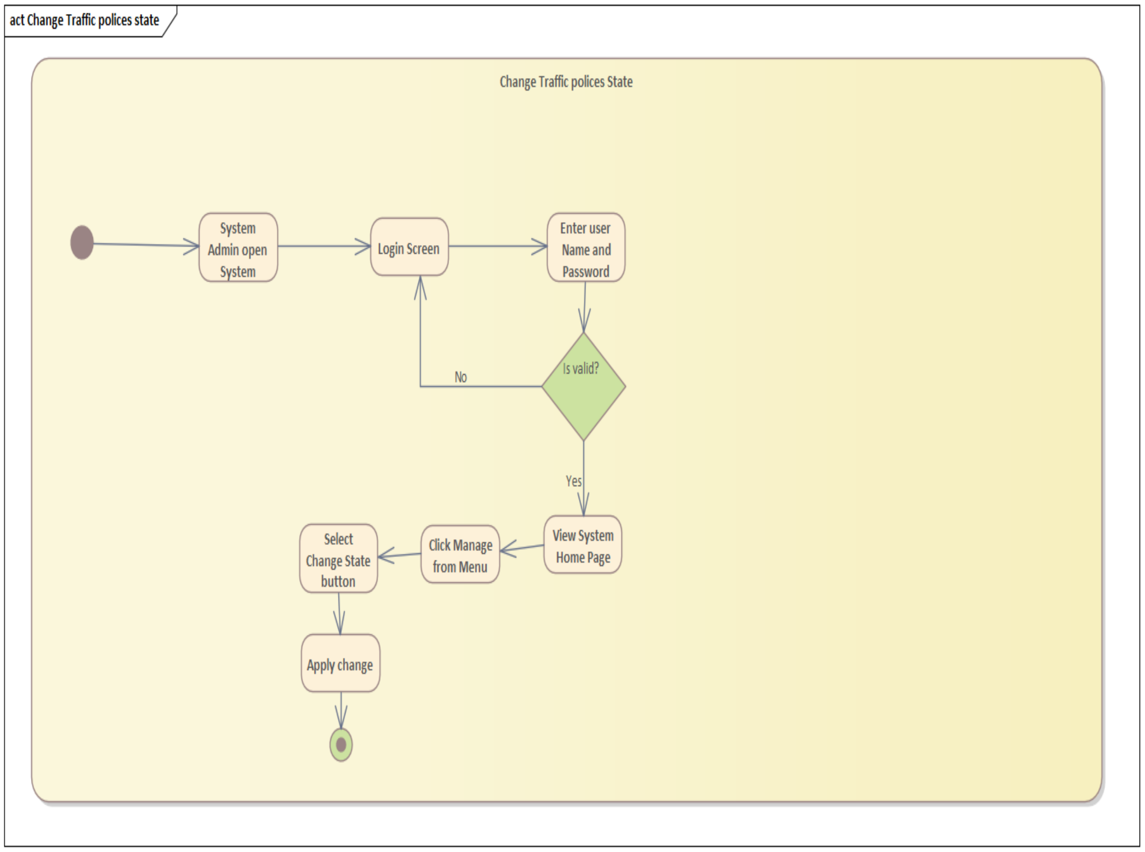

act Change Traffic polices state
Change Traffic polices State
System Admin open System
Login Screen
Enter user Name and Password
Is valid?
No
Yes
View System Home Page
Click Manage from Menu
Select Change State button
Apply change

Figure 3. 19 Activity Diagram of Change Traffic Polices State

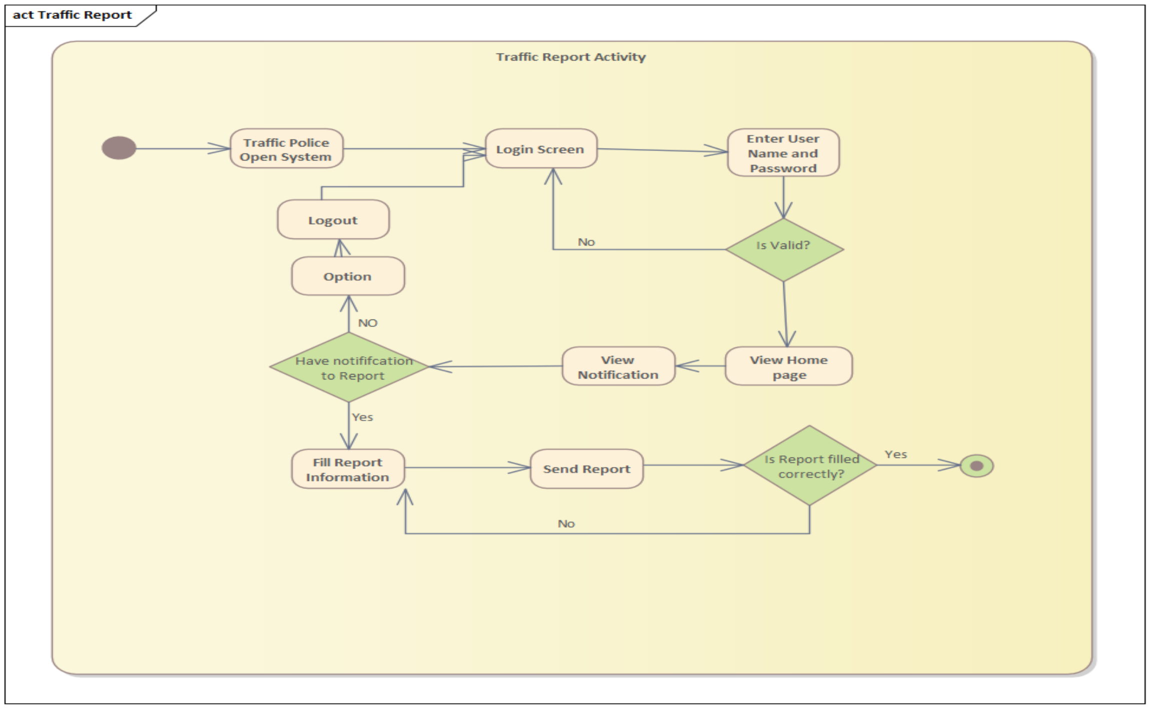


Figure 3. 20 Activity Diagram of Traffic Polices Report and View notification

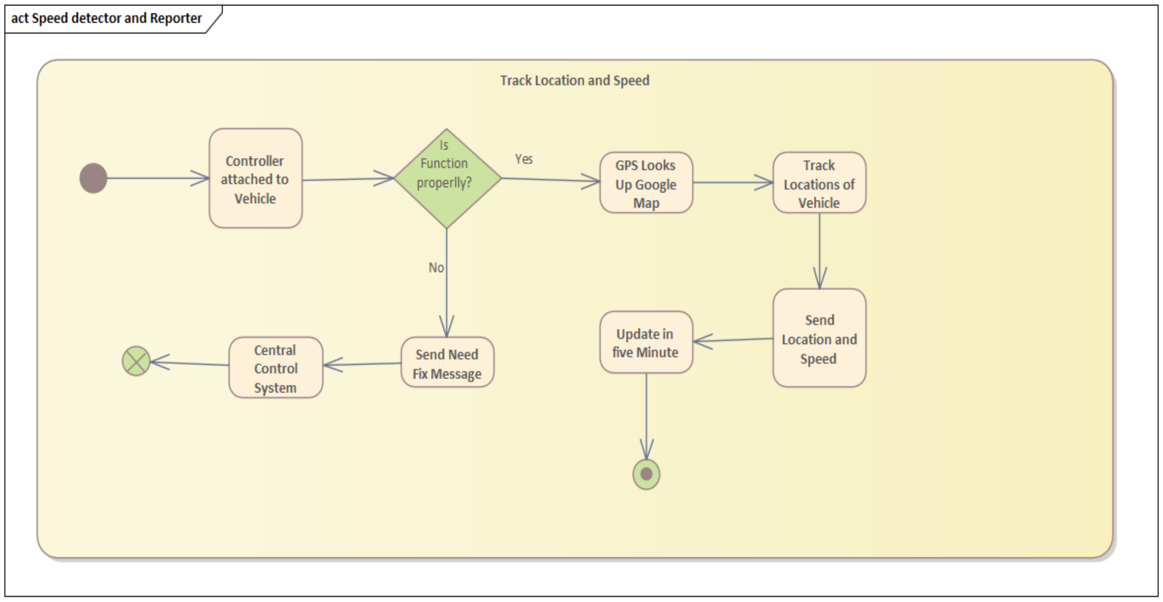


Figure 3. 21 Track Location and Speed of Vehicle

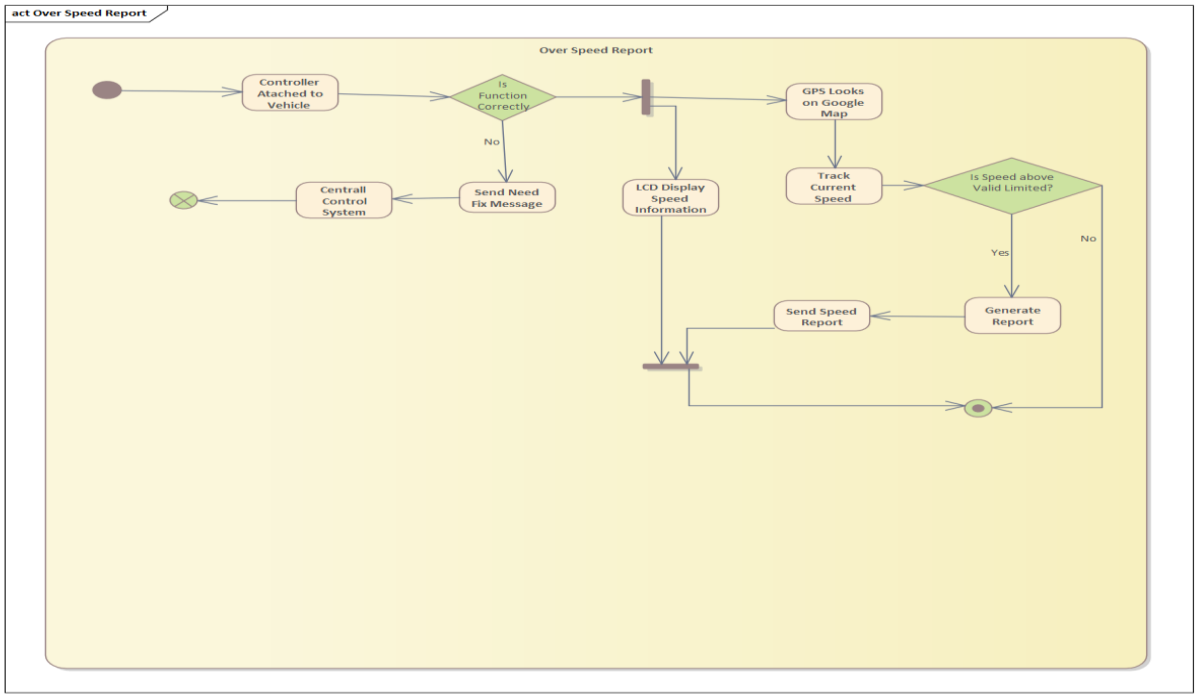


Figure 3. 22 LCD displayer and Above limit speed Reporter

3.5.5. State chart diagram

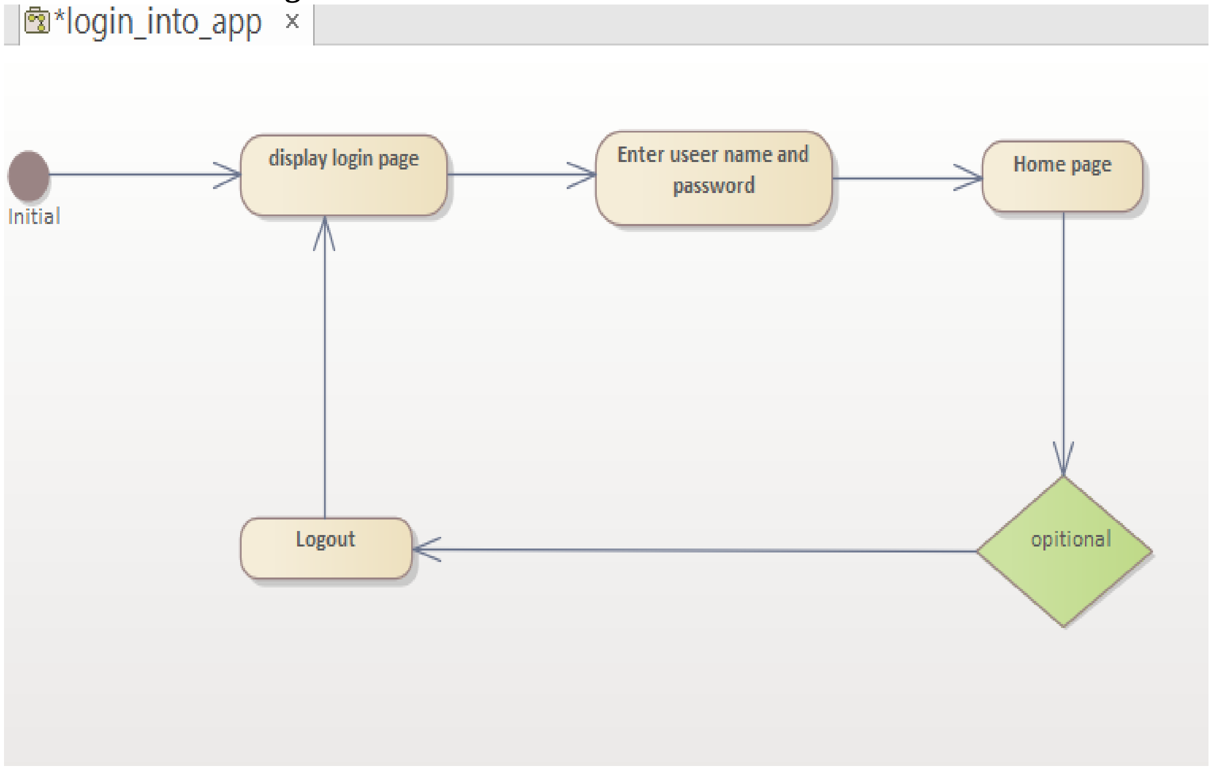


Figure 3. 23 state chart diagram of traffic police login to android app

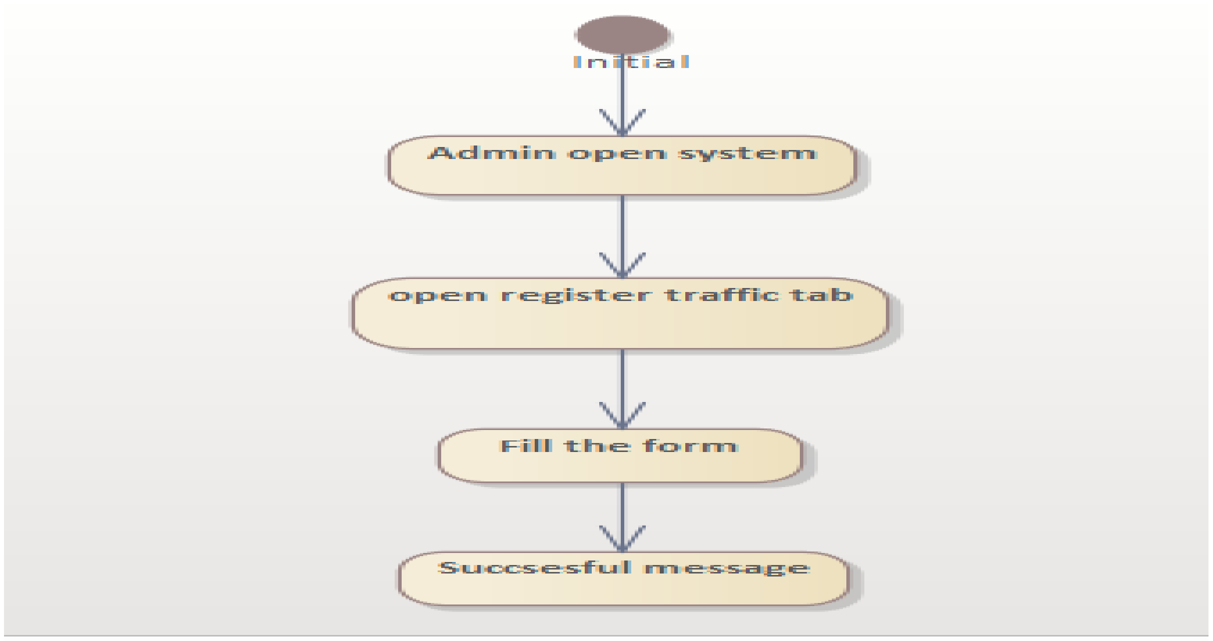


Figure 3. 24 State chart diagram of Traffic Polices Registration

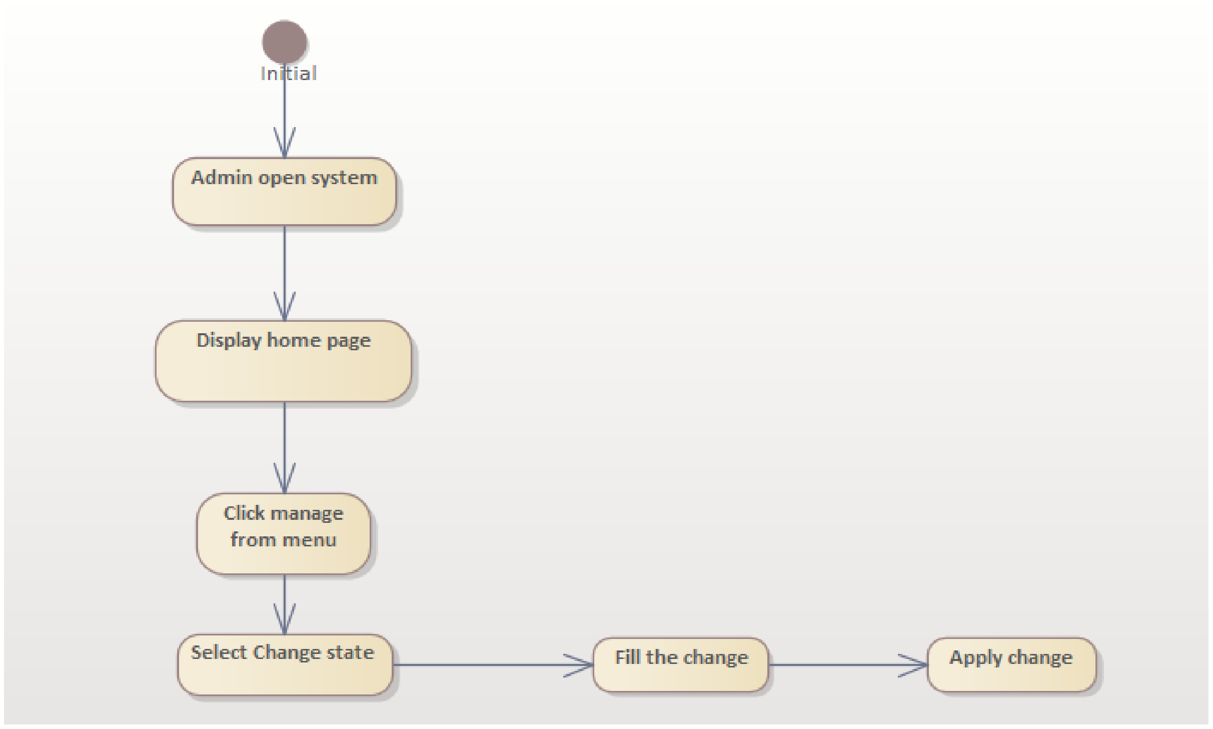


Figure 3. 25 State chart diagram of Change Traffic Polices State

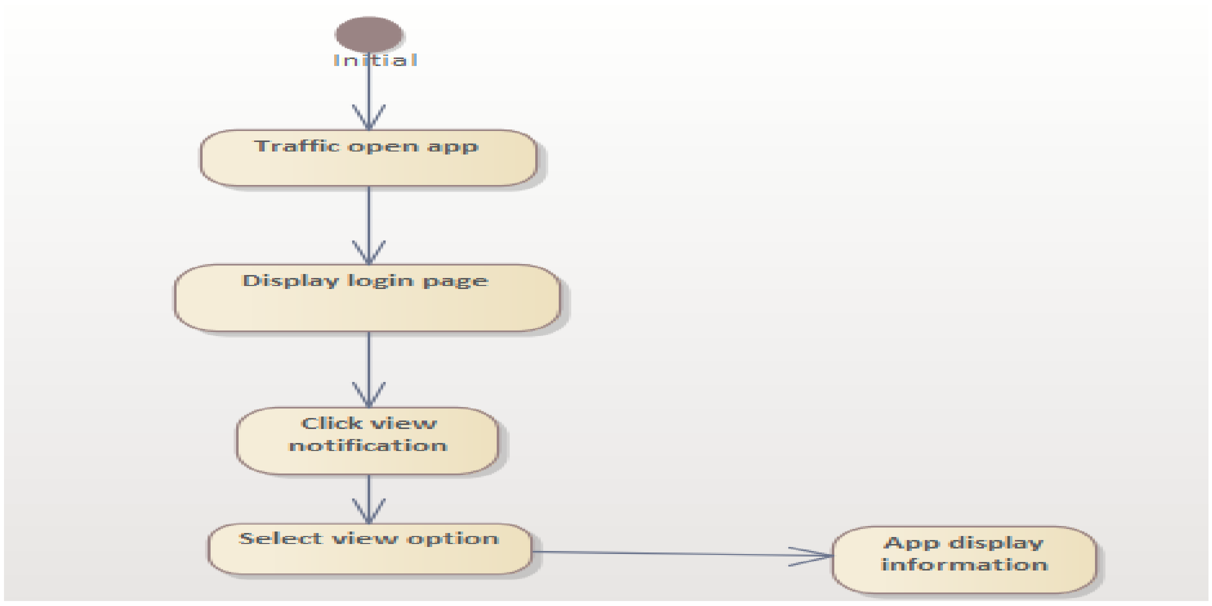


Figure 3. 26 State chart View notification

CHAPTER FOUR

SYSTEM DESIGN

4.1 Overview of the system

Real-time vehicle speed detector and notifier is the system that works with GPS and GMS technology. It identifies roads given speed limit, and if the speed with which the vehicle is traveling is more than the specified speed limit, that data will be sent out to the central control system which consists a website that helps to monitor vehicle movement, then it goes from central control system to the nearby traffic police through the mobile app traffic polices use. After taking actions traffic polices will then give feedback to the central control system. In this chapter, system design of the above-described procedures will be presented.

4.1.1 Purpose of the system design

Purpose of this projects design system is to design and track the necessary information required to effectively define the proposed system architecture and give guidance on the architecture of the system to be developed. Complete architectural overview of proposed system will be designed so as to make easy the implementation parts. It provides the complete architectural overview of the proposed system. It is intended to capture and express the significant architectural decisions which have been made on the system as well as to obtain the information needed to manage all citizens of the country.

4.1.2 Design goal

Design goals are targets for design works. They are agreed upon by developing team and corresponding road authority and users as the criteria for comprising alternatives and evaluating design outcomes. Some of the design goals are discussed below.

Usability

LCD which is a controller part will clearly display to the driver and Traffic polices, the speed limit allowed for the road the driver is on, vehicle current location, and warning incase the sated speed limit is passed. It is attached to the place where the driver can easily see.

Compatibility

The app that traffic polices use receives detail information about the vehicle that has broken the sated speed limit. It also has capacity of sending back the report to the central control system. It is compatible.

Performance

The system should respond fast with high throughput. It should have to perform the task quickly possible as possible such as generating report and receiving, viewing vehicle overspeed record history and vehicle detail, sending alarm and receiving it at the EVSC system. The system performs its task within a user acceptable time and space. Depending on the strength of available network the system should response in short period of time and to do work efficiently the storage capacity of the processor should have to be more than 2GB RAM.

Dependability

- ✓ Robustness: - the web-based system part, that mainly use a menu driven entry would not be an input problem by the user side. However, for the server side there might be an error during the process of entering a data. In this time the system will provide an error page to the admin or Traffic police who is trying to input data, and the system will continue without failure or crush.
- ✓ Availability: - as long as there is an internet connection, the system will be available 7 days a week and 24 hours a day.
- ✓ Security: - EVSC should be secured, i.e., not allow other users or unauthorized users to access data that has no the right to access it.
- ✓ Reliability: EVSC's whole system largely depends on the functionality of the controller. Controller can be affected by accident that happens to the vehicle. Other than that, it has ability to continuously communicate with the satellite and process assigned functionalities accordingly.

Maintainability
In time of failure or need modification the system, need to be maintainable. The following maintenance criteria can be achieved easily.

- ✓ Extensibility: - if it is needed to add new functionality to the EVSC web-based system and the EVSC mobile application, it can be achieved by easily making a separate page and integrate this page with the existing system. If it is needed to add new functionality to the controller, it can be done by connecting the device that has the required functionality to the controller according to their pin connection.
- ✓ Modifiability: - if in the system, some functionality requires to be modified, this modification must be done specifically to that function or page without affecting the overall system organization.
- ✓ Portability: - the system is developed to be viewed and retrieved from any web browser regardless of their version and platform it resides in it. Controller can easily be configured with any vehicle body.
- ✓ Readability: - the system code can be viewed by clicking on the current web page and choose "view the source code" option.

End user

The system has simple and understandable graphical user interface such as forms and buttons, which have all descriptive names. Upon users filling forms, if they have made any error, the system will notify them to correct it accordingly. All the interfaces, forms and buttons are written or designed in a simple language so that the user can access it without any difficult.

4.2 Proposed system architecture

Our systems three tier architecture is shown below.

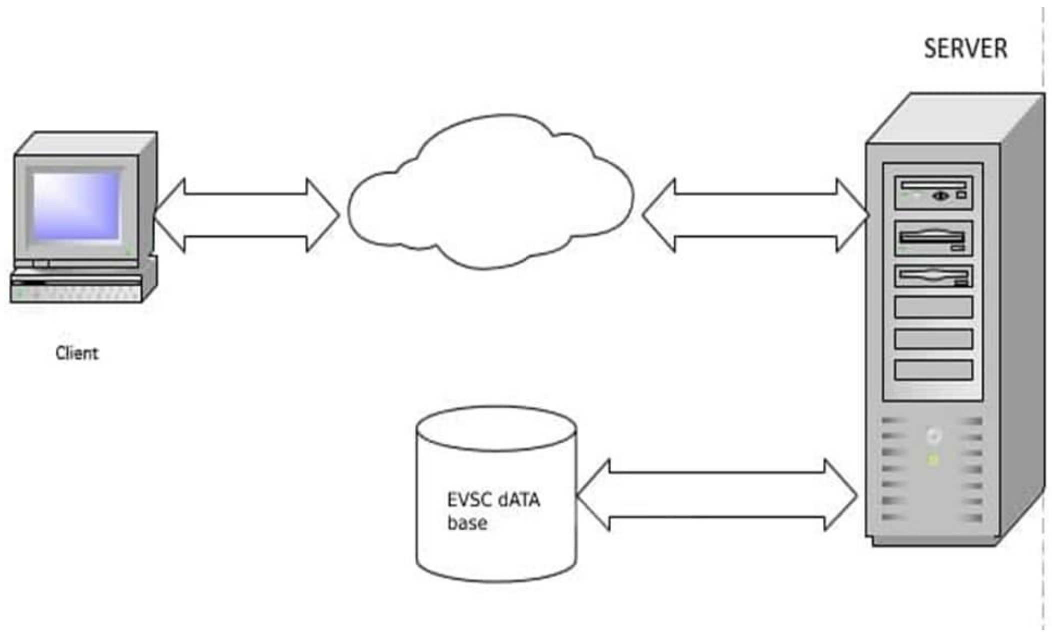


Figure 4. 1 system architecture

4.2.1 System process

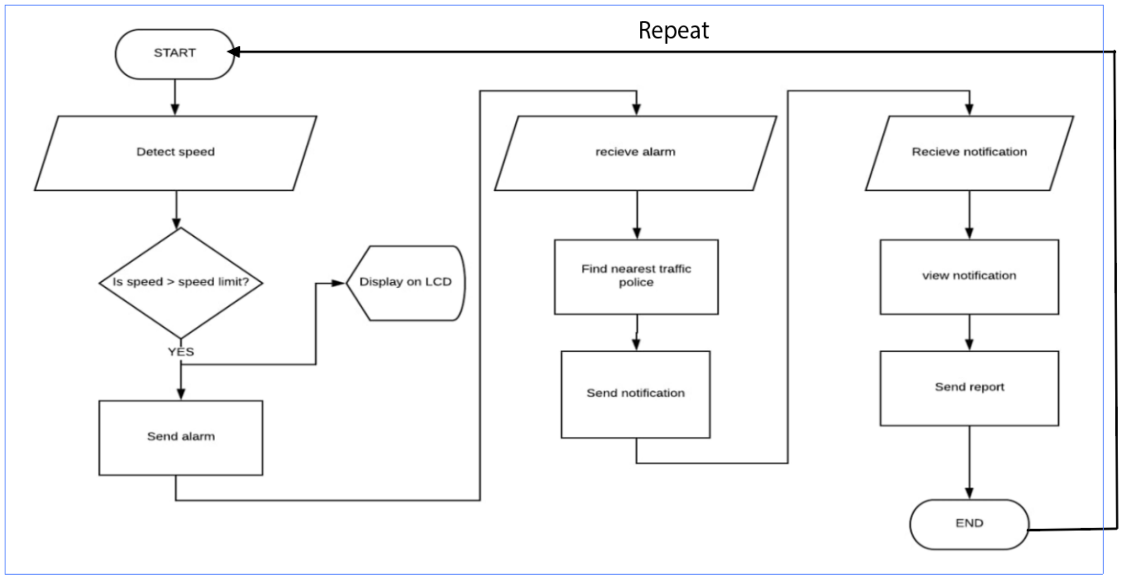


Figure 4. 2 flowchart of our projects system process

4.2.2 Subsystem decomposition

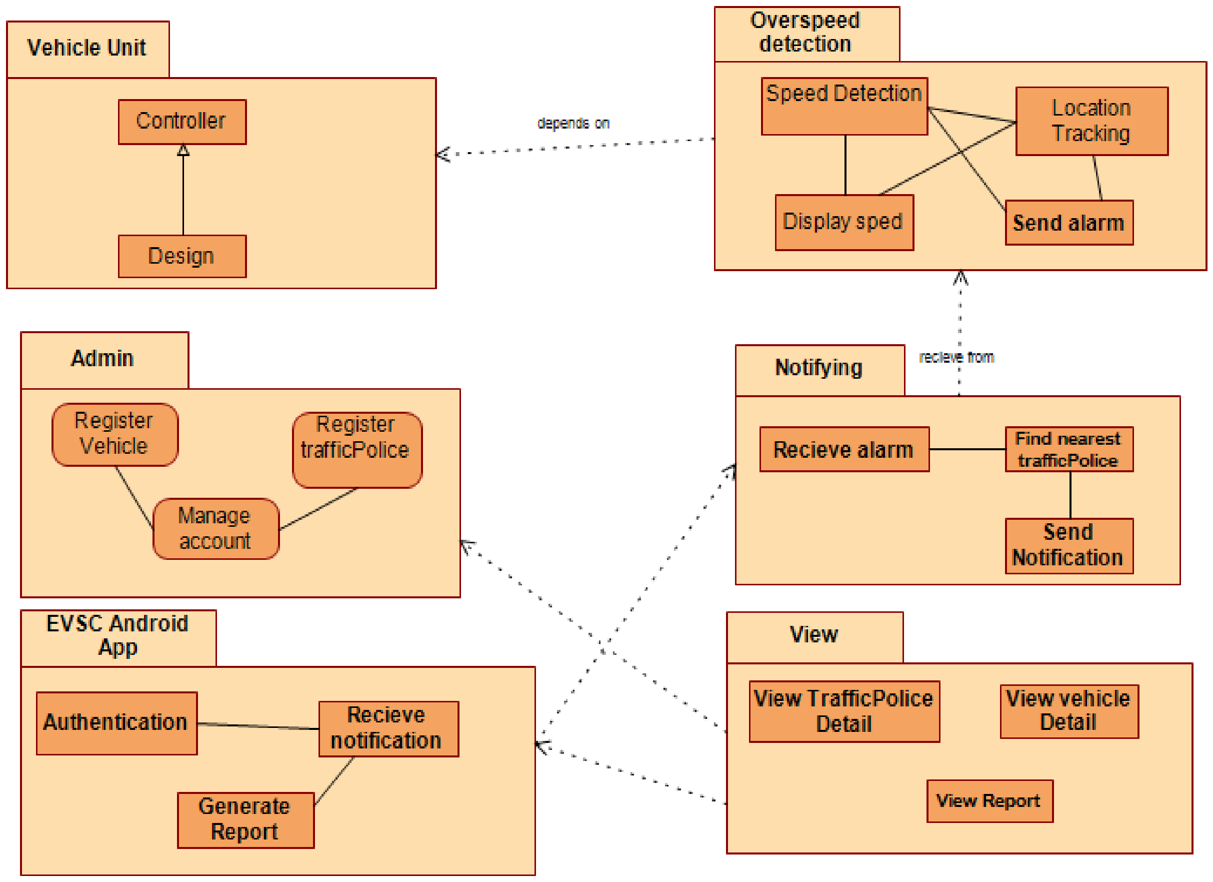


Figure 4. 3 subsystem decomposition diagram

Description of subsystem decomposition

Subsystem	**Purpose**	**Class**
Vehicle unit	It is responsible to design the controller which is installed in the Vehicle that detect the over speed event and send the detected event information to the control database server.	
Overspeed Detection	This subsystem is responsible to detect/extract the speed of vehicle and compares the extracted speed and the allowable speed, if the extracted speed is larger than the allowable speed it extracts the current position of the vehicle from the GPS. Then send the Over speed event and location in to the EVSC system via GSM modem. It also displays the information on LCD display.	Vehicle, Vehicle data record

Admin	Responsible for registering vehicle and traffic police and to update their status Responsible for managing the account.	User account, vehicle, Traffic, Vehicle data record
Notifying	Is responsible to receive the notification, to find nearest traffic police and to send notification to traffic police. And store the information on the database.	Traffic, Vehicle data record, report,
View	Allow admin to see vehicle and traffic police data and also to see report generated.	Report
EVSC android app	Allow traffic police to login and view notification. It also allows to generate report.	Traffic

Table 4. 1 description of subsystem decomposition

4.2.3 Hardware software mapping

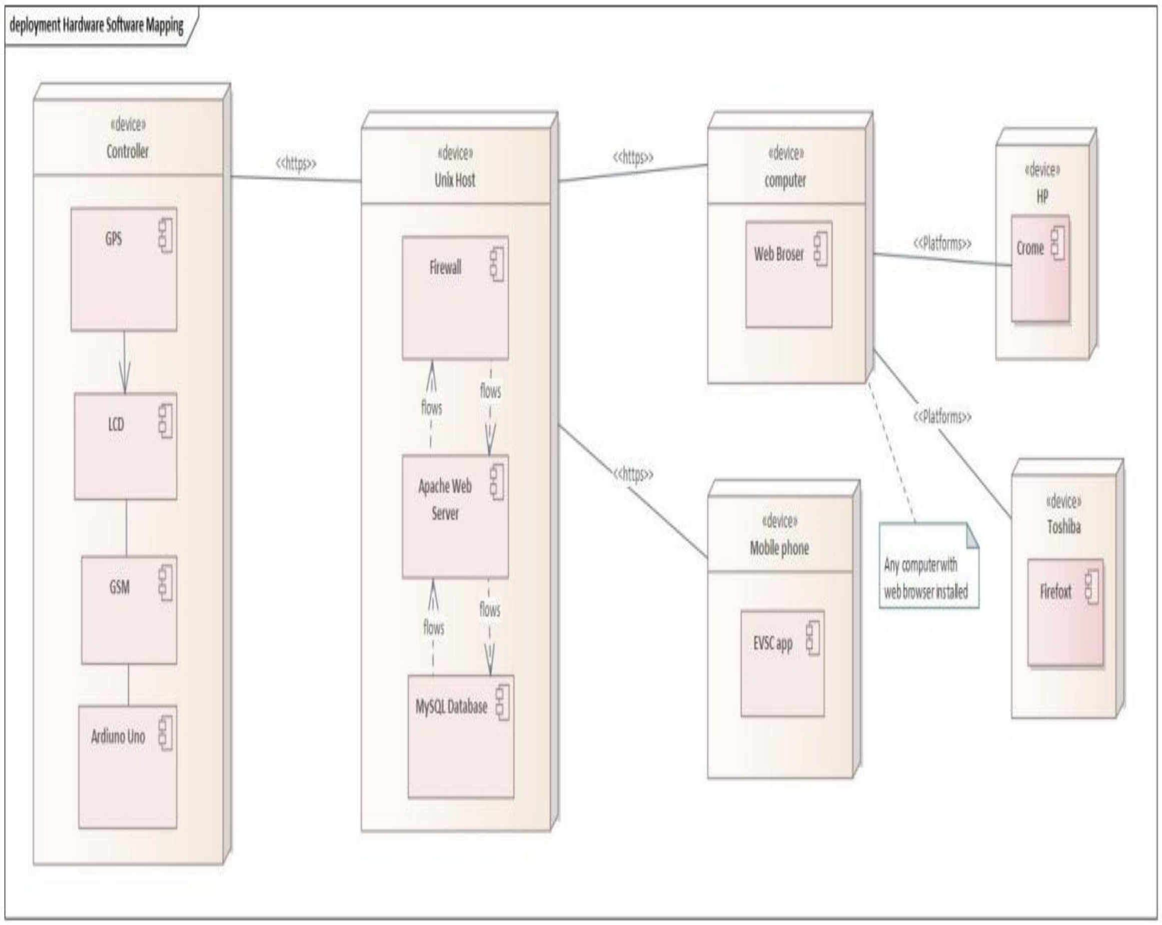
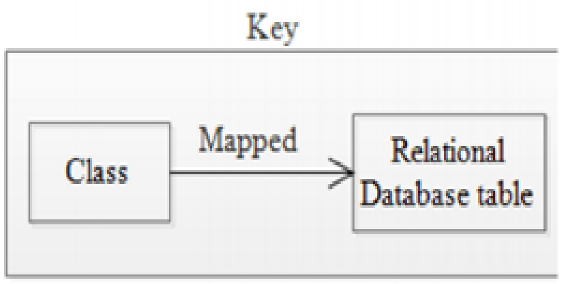


Figure 4. 4 hardware software mapping diagram

4.2.4 Persistent data management

The purpose of this section is to show the mapping of the objects/classes of the system, identified during the analysis stage, in to the corresponding relational database.



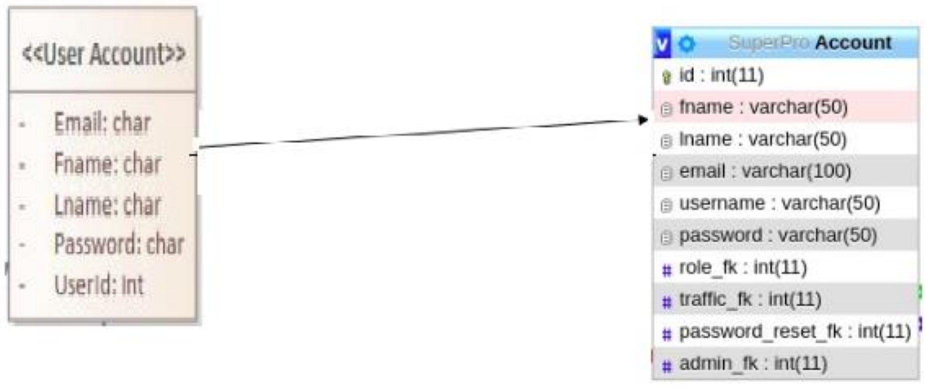


Figure 4. 5 object for account

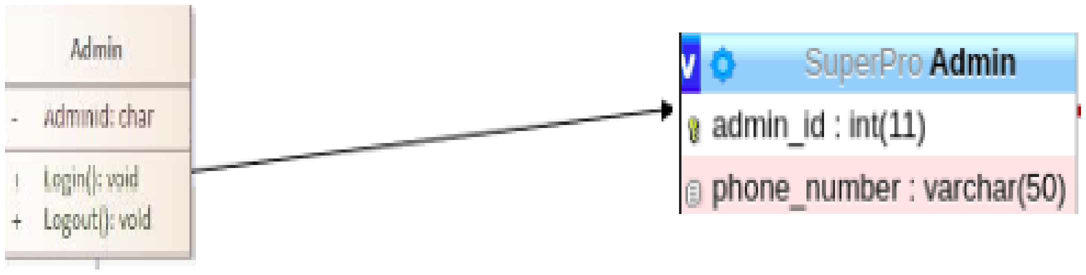


Figure 4. 6 object for admin

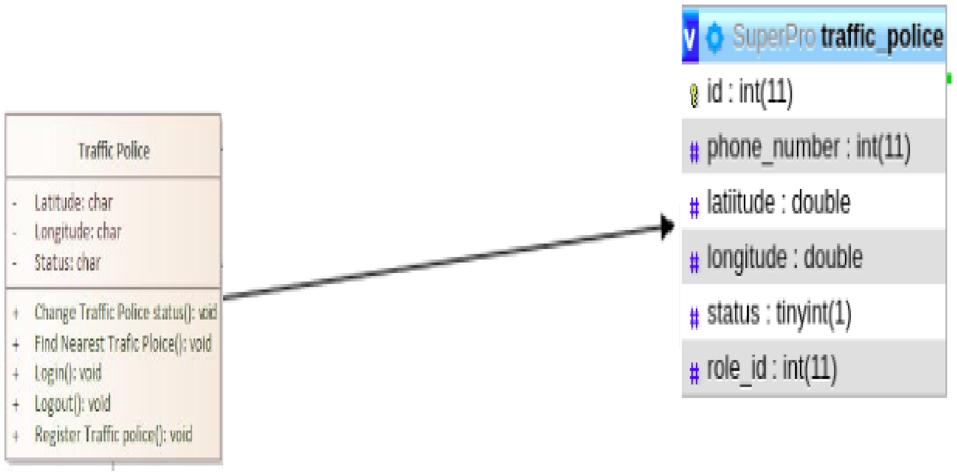


Figure 4. 7 object for traffic police

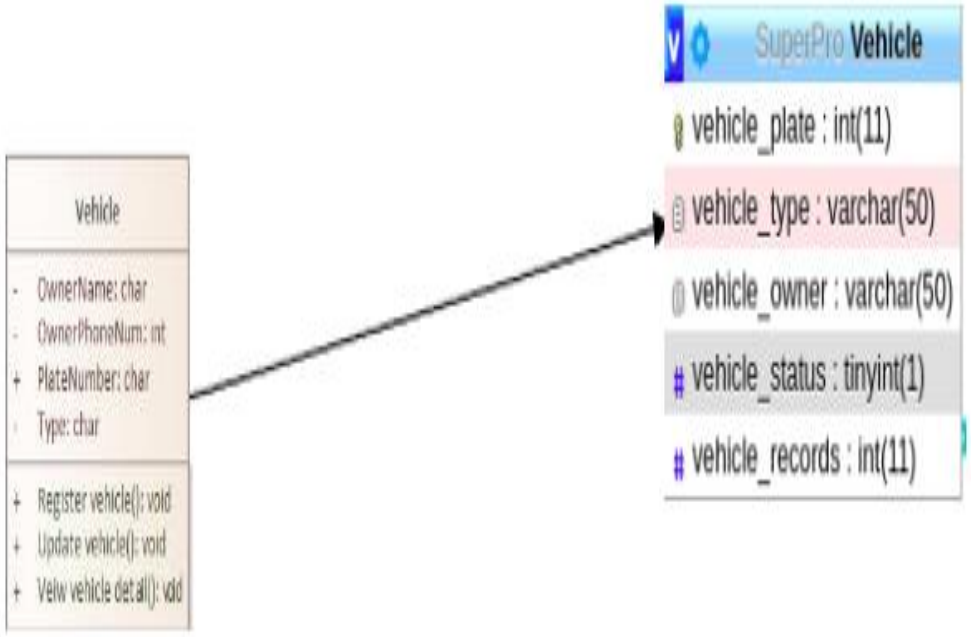

Vehicle
- OwnerName: char
- OwnerPhoneNum: int
+ PlateNumber: char
Type: char
+ Register vehicle(): void
+ Update vehicle(): void
+ Veiw vehicle detail(): void
SuperPro Vehicle
vehicle_plate : int(11)
vehicle_type : varchar(50)
vehicle_owner : varchar(50)
vehicle_status : tinyint(1)
vehicle_records : int(11)

Figure 4. 8 object for vehicle

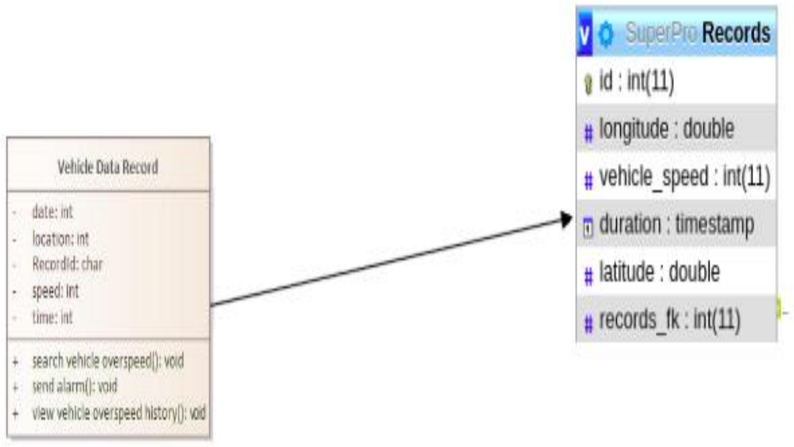


Figure 4. 9 object for vehicle data records

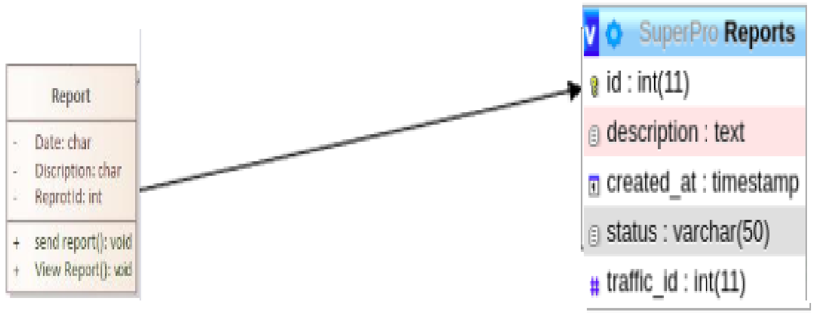


Figure 4. 10 object for reports

4.2.5 Component diagram

Component diagrams provide a simplified, high-order view of a large system. Classifying groups of classes into components supports the interchangeability and reuse of code. This document will provide how these components are composed and how they interact in a system.

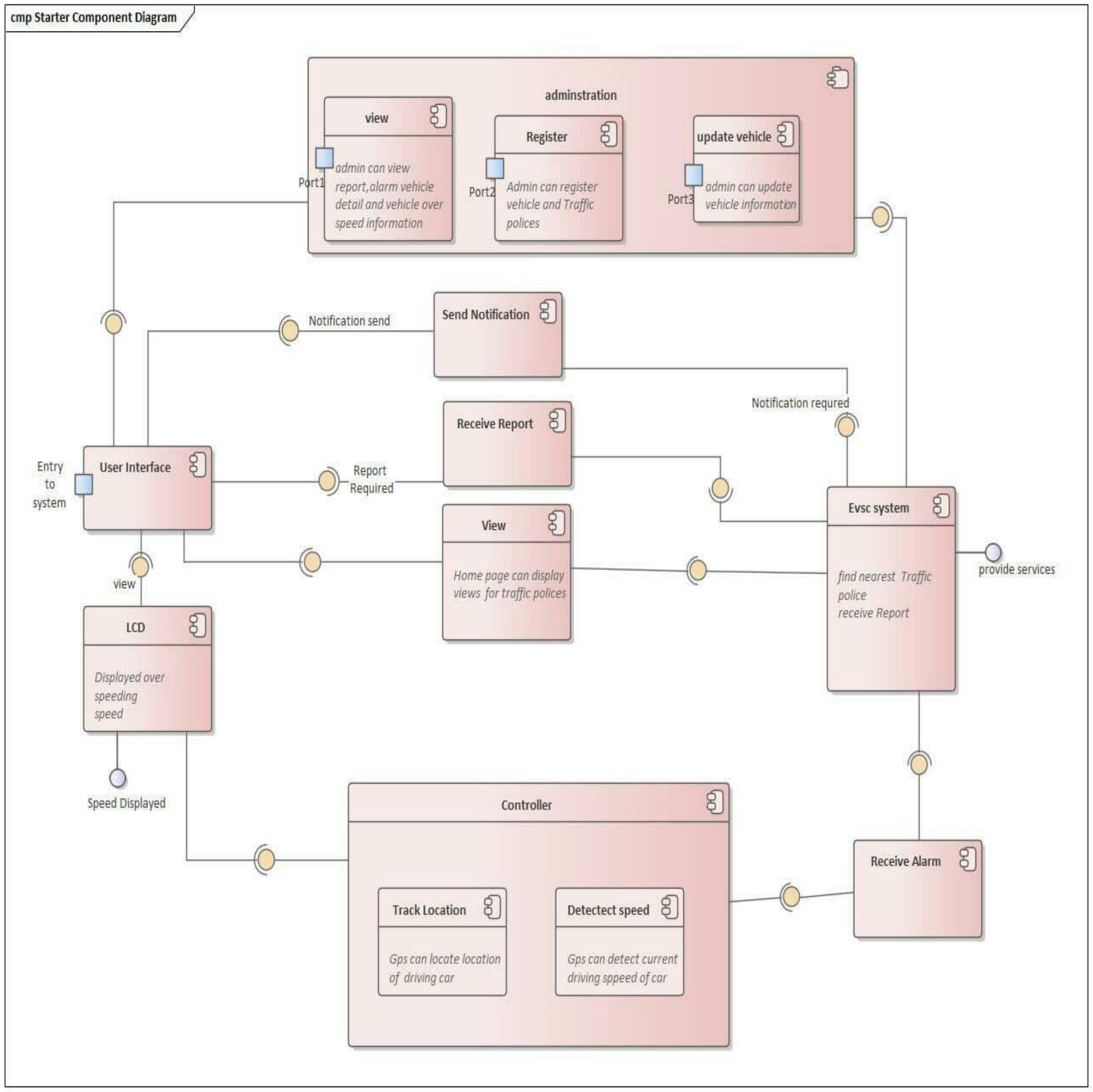

cmp Starter Component Diagram
adminstration
view
admin can view report,alarm vehicle detail and vehicle over speed information
Port1
Register
Admin can register vehicle and Traffic polices
Port2
update vehicle
admin can update vehicle information
Port3
Send Notification
Notification send
Notification requred
Receive Report
Report Required
User Interface
Entry to system
View
Home page can display views for traffic polices
Evsc system
find nearest Traffic police receive Report
provide services
view
LCD
Displayed over speeding speed
Speed Displayed
Controller
Track Location
Gps can locate location of driving car
Detectect speed
Gps can detect current driving sppeed of car
Receive Alarm

Figure 4. 11 Evsc component diagram

4.2.6 Database design

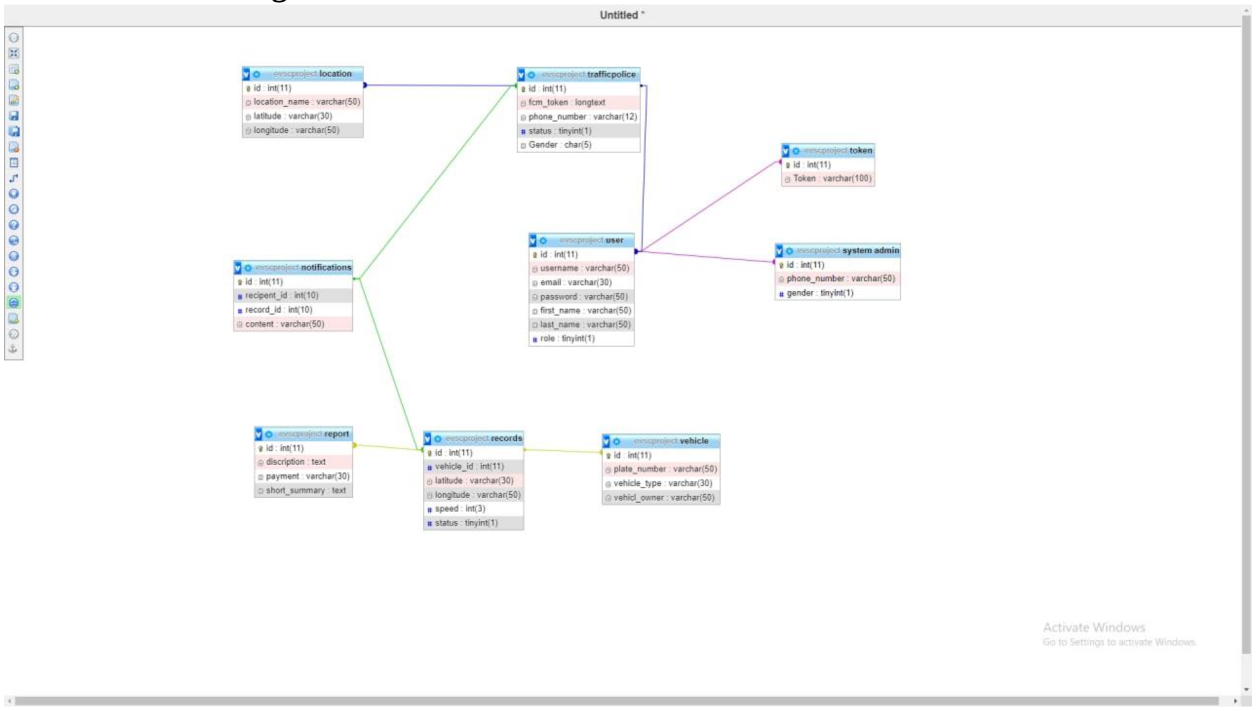


Figure 4. 12 database design

4.2.7 Access control

In our system there are different actors with their respective access privilege. The following part shows actors with their privilege.

Admin: -has the following privilege

- Login
- Logout
- Register Vehicle
- Register Traffic Police
- View Alarm
- View Vehicle detail
- Change Traffic Police Status
- View Report
- Update Vehicle

Traffic Police: -has the following privilege

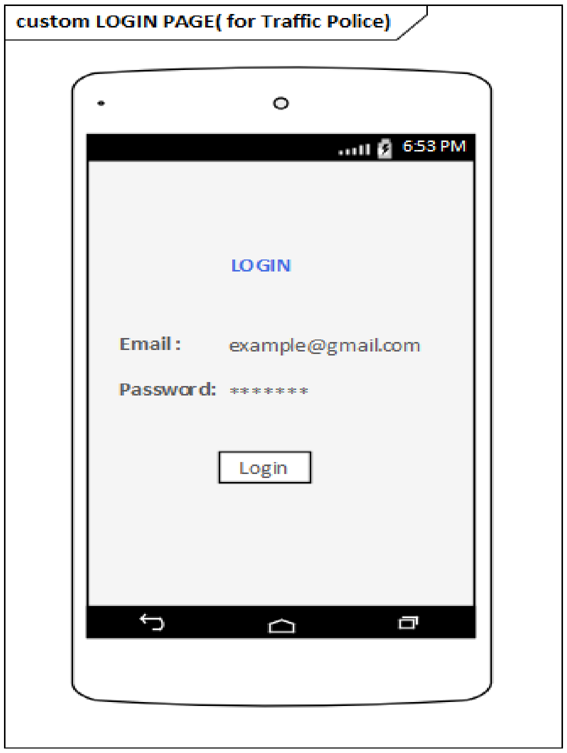
- Login
- Logout
- Receive Notification
- Search Vehicle Over Speeding History

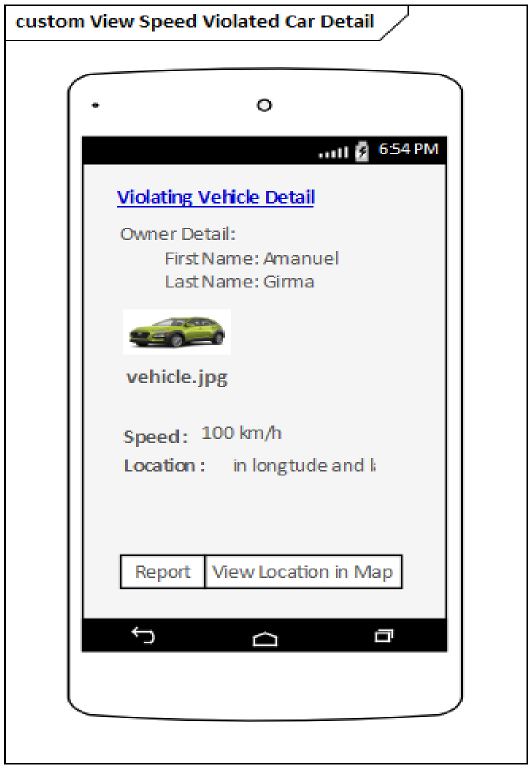
- Send Report

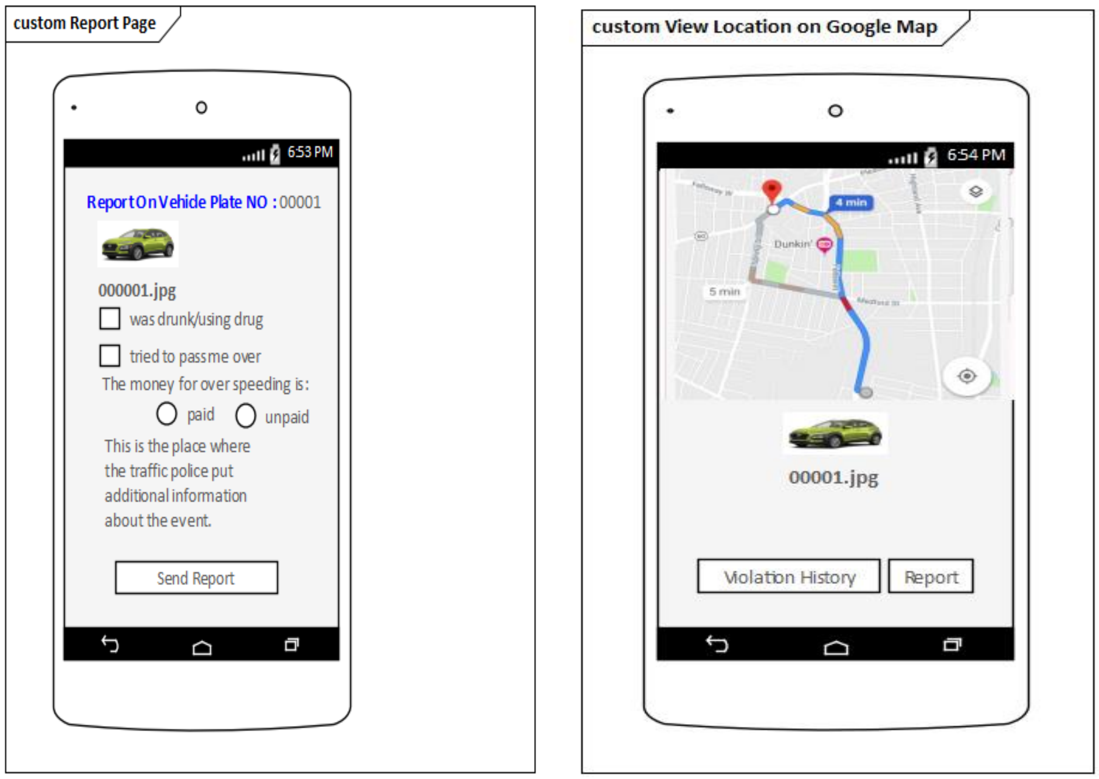
Controller: -has the following privilege

- Detect Speed
- Track Location
- Send Alarm
- Display Speed

4.2.8 User interface design of android app






custom Report Page

6:53 PM

Report On Vehicle Plate NO : 00001

000001.jpg

☐ was drunk/using drug

☐ tried to passme over

The money for over speeding is:

○ paid ○ unpaid

This is the place where the traffic police put additional information about the event.

Send Report

custom View Location on Google Map

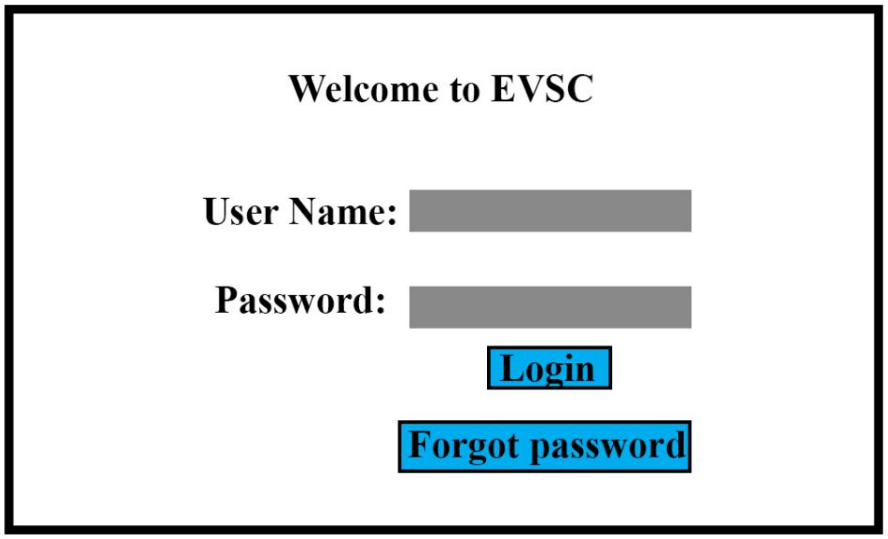
6:54 PM

00001.jpg

Violation History | Report

Figure 4. 13 interface design of android app

EVSC central system user interface for login page


Welcome to EVSC

User Name:

Password:

Login

Forgot password

Figure 4. 14 Evsc web login interface design

CHAPTER FIVE

IMPLEMENTATION

5.1 Overview

Implementation is the process of integrating the system functionality. Our project implements the functional and non-functional requirements of the system. GPS, GSM, Arduino Uno, and LCD display will be configured and packed as one device. All the three subparts of our project will be done one by one. In this chapter, implementation of our projects will be discussed.

5.2 Coding Standard

We use a coding standard that forces consistency and uniformity throughout our codes that helps our team in the following ways:

- ✓ Improve readability, maintainability and reduce complexity of code
- ✓ Improving efficiency of the programmer in the long runs
- ✓ To have a program that have a consistent and uniform format even when they are written by different developers
- ✓ To make the code elegant and efficient. To have a program that makes it hard for bugs to hide, to have dependencies that are minimum to easy maintenance and error handling.

Here are the coding standards that we followed in the implementation of our project:

❖ **Meaningful naming**

Names are everywhere in software. We name our variables, our functions, our arguments, classes, and packages. We name our source files and the directories that contain them. Since they are the fundamental part of a software, they need a standard that provoke good naming. The following are some of the standards we use for creating good names.

.1. **Use Intention-Revealing Names**
The name of a variable, a function, a class or any name within the software should tell why it exist, what it does and how it is used. Choosing names that reveal intent makes it much easier to understand and change code.

.2. **Avoid Disinformation**
Avoid using names that are specific to particular thing. For example, it is not recommended to refer to grouping of accounts to an account List unless it is actually a list. Otherwise, it is better to refer to it as accounts. Another example of misinformative naming would be use of lowercase L and uppercase o as variable names because they resemble the constants one and zero respectively.

.3. **Make Meaningful distinction**
Avoid adding different numbers and noise words to different names within the same scope of the program. Instead, make a meaningful distinction between them.

.4. **Use Pronounceable Names**

The names should be pronounceable.

.5. **Avoid Encoding**

Interface and Implementation: we generally need to avoid encoding them. But if we, must we will encode the implementation.

.6. Class Names

Classes and Objects should have noun or noun phrase names like Customer, Account and Address Parser.

.7. Method Names

Methods should have verb or verb phrase names like post Payment, delete Page, or save.

Accessors, mutators, and predicates should be named for their value and prefixed with get,

set, and is according to the JavaBean standard.

.8. Camel Case Naming, Using Underscore

Camel case naming should be used in programs that will be written in Kotlin, Java and C for python underscores should be used to separate the words.

❖ **Functions**

.1. Small

Functions should be small. If it is possible, make them smaller than 20 lines.

.2. Blocks and Indenting

Functions should not be large enough to hold nested structures. Therefore, the indentation level of a function should not be greater than one or two.

.3. Switch Statement

Don't use switch statement in functions since they make code long and increase the level of indentation of the function. Alternative to switch statement are polymorphism.

.4. Function Arguments

The ideal number of arguments is zero. Next comes one followed closely by two. Function argument more than two complex to handle.

Avoid passing Boolean values as an argument as much as possible.

❖ **Comments**

The code itself should be self-descriptive and comments should apply to describe why the code is used instead of how it works. And use commenting as a guide to keep the logic straight while writing the code.

.1. Copyright and Authorship

copyright and authorship statements are necessary and reasonable things to put into a comment at the start of each source file.

.2. Informative comments

Sometimes use comment to explain codes say the return value of an abstract method.

5.3 Implementation Detail

To develop Real time vehicle overspeed detecting and notifier system we used different tools and technologies. These tools include hardware and software tools for both client side and server side.

5.3.1 Software

5.3.1.1 Front end/Client Side

In our project there is three implementations, front end, back end and hardware. Front-end is also referred to as the client-side. This is part of application where user interfaces machine operates. because of users have different machines with different kind of capacity client execution performances. client performances typically revolve around reducing the overall size of webpages. This includes the size of the files and perhaps more important, the number of them. Django-python framework which is frontend and back-end framework to build SAP (single page app) ability to load faster without hitting the server every time the user interacts with the application. our project Architecture, there may be many layers between the hardware and end user. The frontend is abstraction, simplifying the underlying component by providing a user-friendly interface, while the backend usually handles data storage and business logic.

Frontend/client-side divides in to two. The traffic polices and the admin.

- ✓ The traffic police front end is where used by traffic police to view notification and to send report.
- ✓ The admin front end/client-side is used by the server admin to view register and view traffic, vehicle details and to register traffic and vehicle.

Django support MVT (Model-View Template) architecture. MVT is a software design pattern for developing a web application.

- ❖ Django Model: - Model is going to act as the interfaces of our data.it is responsible for maintaining data. This is the database part used to make the interfaces between the models created and database server.
- ❖ Django View: - The view is user interfaces-viewed in the browser when user render a website.it is represented by HTML/CSS/JavaScript and Jinja files
- ❖ Django Template: -a template consists of static parts of the desired HTML output as well as some special syntax describing how dynamic content will be inserted.

5.3.1.2 Server side

This is the part of the application where the client computers request services of the server display the results the server returns. Framework used for the backend Django.

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. It was chosen because of its speed and security.

Other dependencies include: -

- ➢ **Django-rest framework: -** framework is a powerful and flexible toolkit for building Web backend APIs.

- ➢ **Django-rest-auth: -** Authentication Module for Django rest –auth provides easy to use authentication for Django REST Framework The aim is to allow for common patterns in applications that are REST based, with little extra effort; and to ensure that connections remain secure.
- ➢ **Geopy:** - to locate the coordinates of addresses, cities, countries, and landmarks across the globe using third-party geocoders and other data sources
- ➢ **GeoDjango:** is an included contrib module for Django that turns it into a world-class geographic Web framework. GeoDjango strives to make it as simple as possible to create geographic Web applications, like location-based services.

5.3.1.3 Android Application

The mobile application is the tool where traffic police use to make report and accept new alarm from the main EVSC system (Web App) and many other actions.

Android Studio is used to develop the Mobile App. Google suggest using Android Studio to develop android apps, because of its active support and development. Number of other libraries like Jetpack compose developed by googles android team makes the development of app smoother, faster and cleaner than ever preserving the native feel of the app. Java or Kotlin is used to write the programs. And as off 2018 Kotlin becomes the first language for Android Development, which reduces the amount code to write and support rapid development. Some of the different libraries that will be used include:

- ❖ Android Jetpack Libraries: - Android jetpack libraries are very useful for the following reasons:
- ✓ They provide Modern App Architecture
- ✓ They Eliminate Boiler Plate Code
- ✓ They Simplify Complex Tasks
- ✓ Robust Backwards Compatibility
- ❖ Retrofit Library: -Is a Type-Safe HTTP client for android. It is used to access APIs from the EVSC system, Google Maps and other API providers for required services of the app.

5.3.2 Hardware

Our project intends to propose/develop and install the Over Speeding Detector device for vehicle. The vehicles are fitted with over speeding detector device(controller) which has the capability for receiving information, recording and sending the data about the speed of vehicle. The proposed system mainly consists of a microcontroller and GPS and GSM module. Vehicle parameters and location coordinates from the GPS module are fed to the controller. The controller transfers this data to the server with the help of GSM/GPRS technology. Typical vehicle parameters monitored are vehicle location, vehicle speed and time. These parameters are stored in the database on a web server and a webpage is created to display vehicle parameters data.

- ❖ **Microcontroller**

Arduino Uno is a microcontroller board based on 8-bit ATmega328P microcontroller. It is an open-source platform of a microcontroller having different variants of boards depending on the controller chip used. Mainly ATMEGA series 8bit controller chips are used.

Figure 5. 1 Arduino Uno

Arduino Uno Technical Specifications

Microcontroller	**ATmega328P – 8-bit AVR family microcontroller**
Operating Voltage	5V
Recommended Input Voltage	7-12V
Input Voltage Limits	6-20V
Analog Input Pins	6 (A0 – A5)
Digital I/O Pins	14 (Out of which 6 provide PWM output)
DC Current on I/O Pins	40 mA
DC Current on 3.3V Pin	50 mA
Flash Memory	32 KB (0.5 KB is used for Bootloader)
SRAM	2 KB
EEPROM	1 KB
Frequency (Clock Speed)	16 MHz

Table 5. 1 technical specifications of Arduino Uno

Pin Description

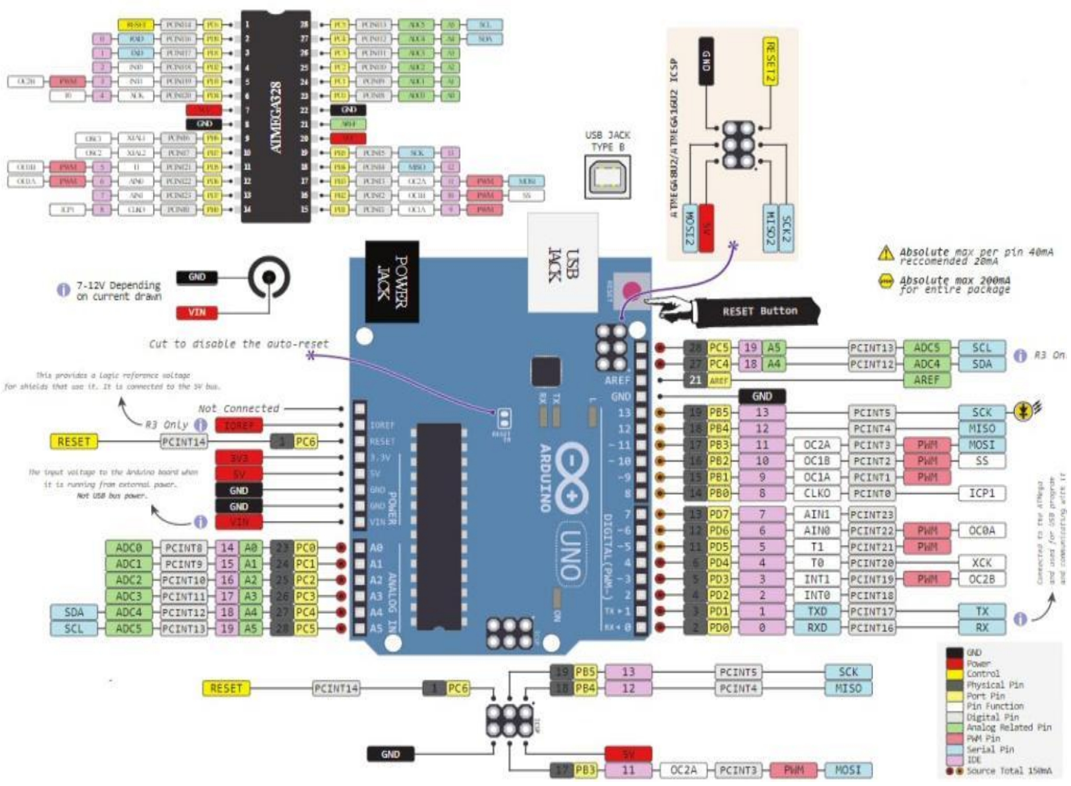


Figure 5. 2 pin description of Arduino Uno

Pin Category	Pin Name	Details
Power	Vin, 3.3V, 5V, GND	1. Vin: Input voltage to Arduino when using an external power source. 2. 5V: Regulated power supply used to power microcontroller and other components on the board. 3. 3.3V: 3.3V supply generated by on-board voltage regulator. Maximum current draw is 50mA. 4. GND: ground pins.
Reset	Reset	Resets the microcontroller.

Analog Pins	A0 – A5	Used to provide analog input in the range of 0-5V
Input/output Pins	Digital Pins 0 - 13	Can be used as input or output pins.
Serial	0(Rx), 1(Tx)	Used to receive and transmit TTL serial data.
External Interrupts	2, 3	To trigger an interrupt.
PWM	3, 5, 6, 9, 11	Provides 8-bit PWM output.
SPI	10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK)	Used for SPI communication.
Inbuilt LED	13	To turn on the inbuilt LED.
TWI	A4 (SDA), A5 (SCA)	Used for TWI communication.
AREF	AREF	To provide reference voltage for input voltage.

Table 5. 2 Pin configuration

GPS Modem

Among the available technologies to be used in measuring the speed of the vehicles, the GPS technology was chosen due to a major reason. It provides multiple and accurate data items, including the speed, location, date and time, which are the pieces of information we are interested in. We use Sunroom's GPS receiver that can acquire GPS signals from 65 channels of satellites and output position data with high accuracy in extremely challenging environments and under poor signal conditions due to its active antenna and high sensitivity.

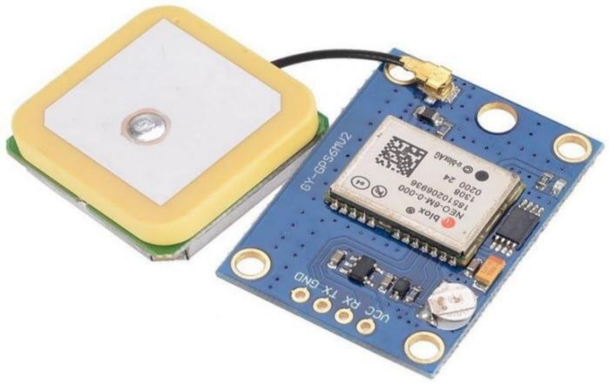


Figure 5. 3 GSM modem

Interfacing the GPS with Microcontroller (Arduino)

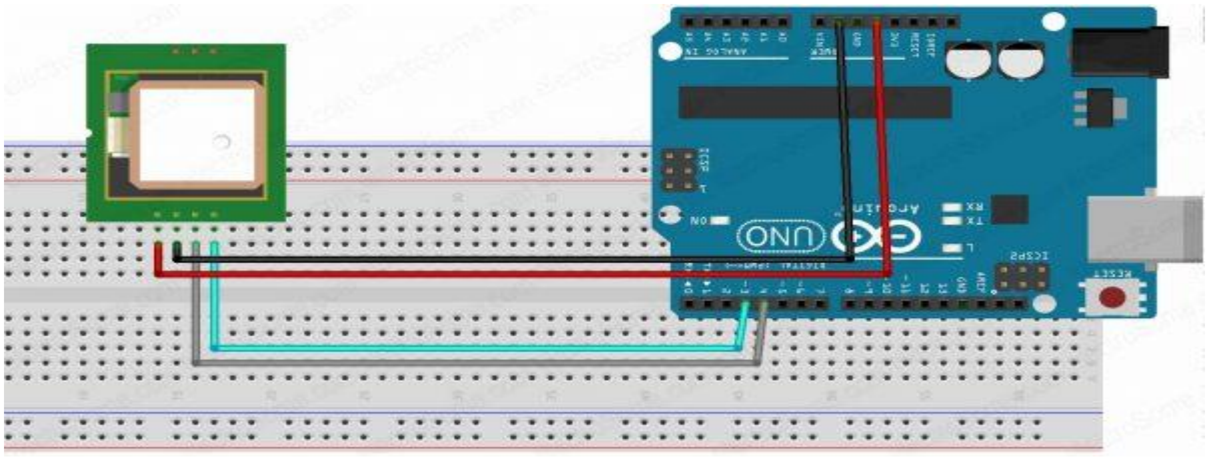


Figure 5. 4 GPS interfacing with Arduino Uno

The connection is as follow:

- ✓ Connect TX pin of GPS module to pin number 4 of Arduino
- ✓ Connect RX pin of GPS module to pin number 3 of Arduino
- ✓ Connect red wire(5V) pin of GPS module to 5V pin of Arduino
- ✓ Connect black wire (GND pin) of GPS module to GND pin of Arduino

GSM Modem

SIM900 enables GPRS connectivity to embedded applications. We can implement http client protocol using SIM900 HTTP function AT Commands. The Hypertext Transfer Protocol (HTTP) is a standard application layer protocol which functions as a request response protocol in between server and client. The GSM/GPRS module uses USART communication to communicate with microcontroller or PC terminal.

Figure 5. 5 GSM module

Interfacing the GSM with Microcontroller

Connecting GSM modem with Arduino is very simple just connect RX Line of Arduino to TX Line of GSM Modem and vice versa TX of Arduino to Rx of GSM modem. Make sure use TTL RX, TX lines of GSM modem. Give 12V 2Amp power supply to GSM modem, use of less current power supply can cause reset problem in GSM modem, give sufficient current to GSM modem.

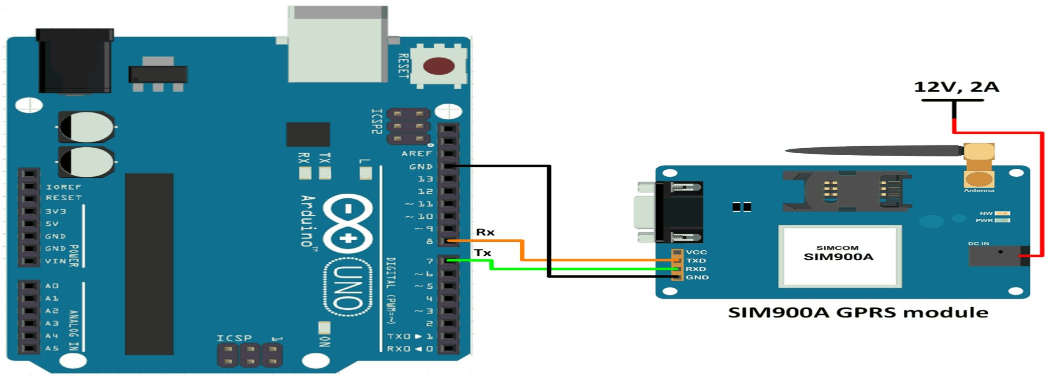


Figure 5. 6 circuit diagram of GSM with Arduino

Here we have used a normal GSM Module with a SIM card for GPRS connection. In this project, GPRS is responsible for sending data to the server**.** GPRS which is a packet based wireless communication service that works with data rate of 56-114kbps and provides a connection to the internet.

For GPRS, we do not need to buy any special module or hardware because GSM already has GPRS facilities inbuilt. We only need to access it by using the same method or AT commands. There are many AT commands already mentioned in the datasheet of SIMCOM SIM900A GSM module.

LCD display

For this project we use **16×2 LCD.** It can display 16 characters in 2 rows. Easy to use with Arduino Uno. Also, operates on the same voltages of Arduino Uno.

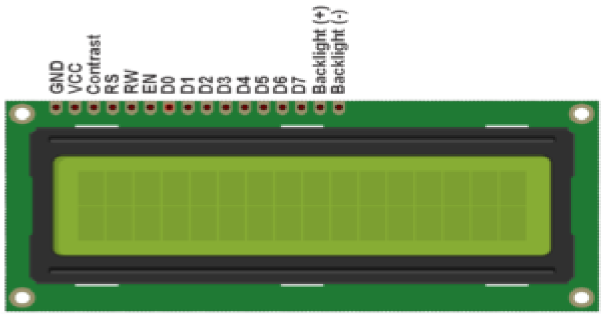


Figure 5. 7 LCD display

- Ground: - Ground pin of the LCD module.
- Pin2(Vcc): - Power to LCD module (+5V supply is given to this pin)
- Pin3(VEE): - Contrast adjustment pin. This is done by connecting the ends of a 10K potentiometer to +5V and ground and then connecting the slider pin to the VEE pin. The voltage at the VEE pin defines the contrast. The normal setting is between 0.4 and 0.9V.
- Pin4(RS): - Register select pin. The JHD162A has two registers namely command register and data register.
- Pin5(R/W): - Read/Write modes. This pin is used for selecting between read and write modes. Logic HIGH at this pin activates read mode and logic LOW at this pin activates write mode.
- Pin6(E): - This pin is meant for enabling the LCD module. A HIGH to LOW signal at this pin will enable the module.
- Pin7(DB0) to Pin14(DB7): -These are data pins. The commands and data are fed to the LCD module though these pins.
- Pin15(LED+): - Anode of the back light LED. When operated on 5V, a 560-ohm resistor should be connected in series to this pin. In Arduino based projects the back light LED can be powered from the 3.3V source on the Arduino board.
- Pin16(LED-): - Cathode of the back light LED.

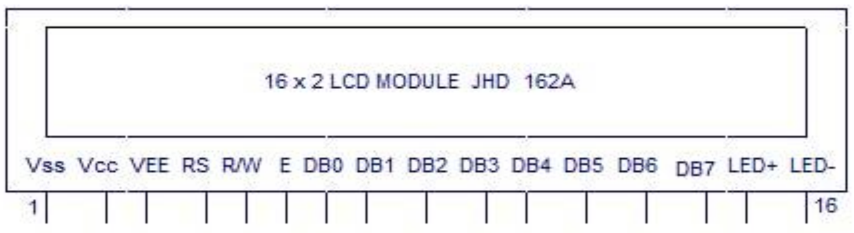


Figure 5. 8 LCD display pin configuration

Connection of Arduino UNO and JHD162a LCD

LCD	**Arduino**
VSS	GND
VCC	5V
VEE	10K Resistor
RS	A0 (Analog pin)
R/W	GND
E	A1
D4	A2
D5	A3
D6	A4
D7	A5
LED+	VCC
LED-	GND

Table 5. 3 Arduino LCD connection

The complete design of controller (Vehicle unit) is shown bellow

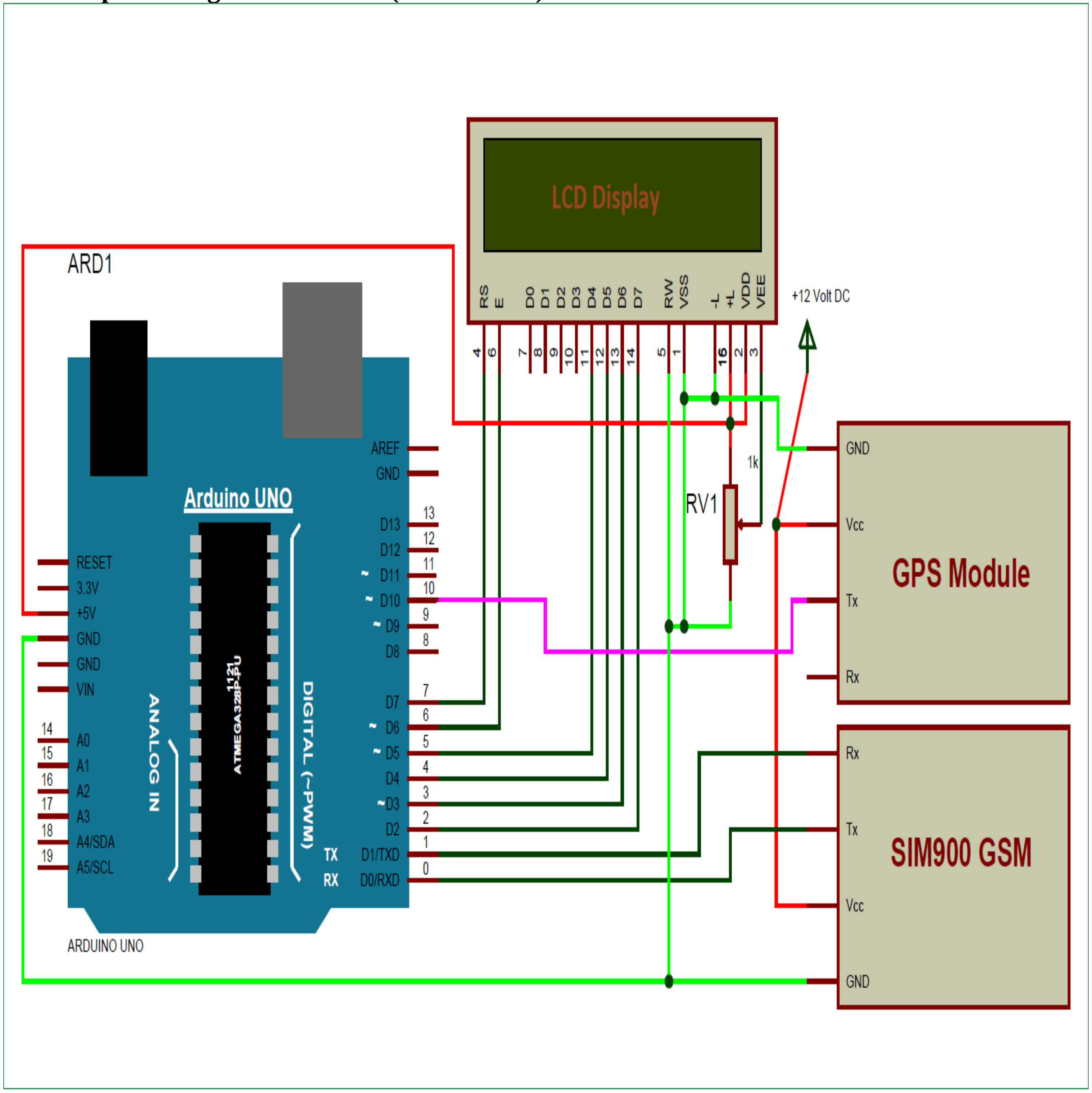
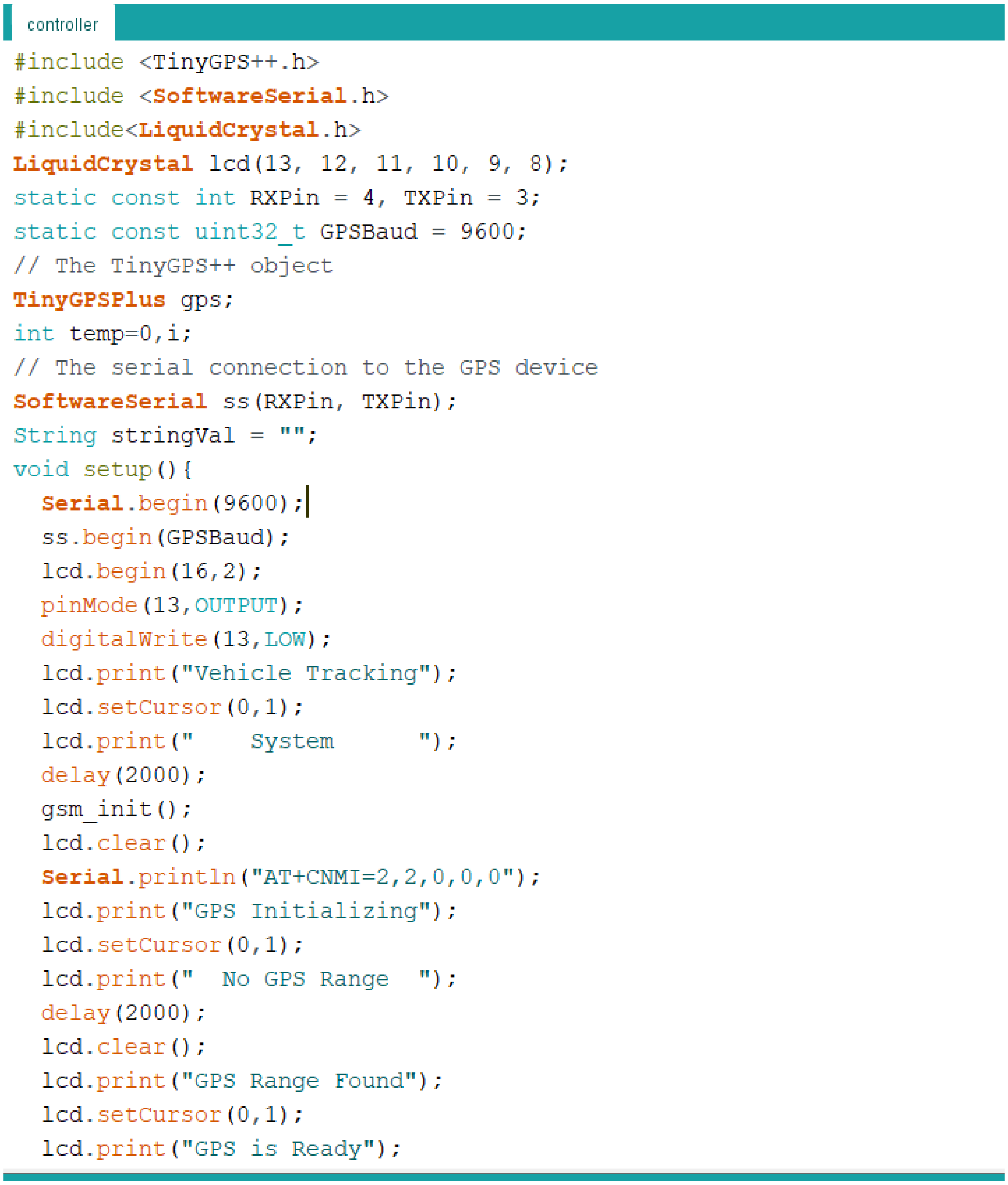


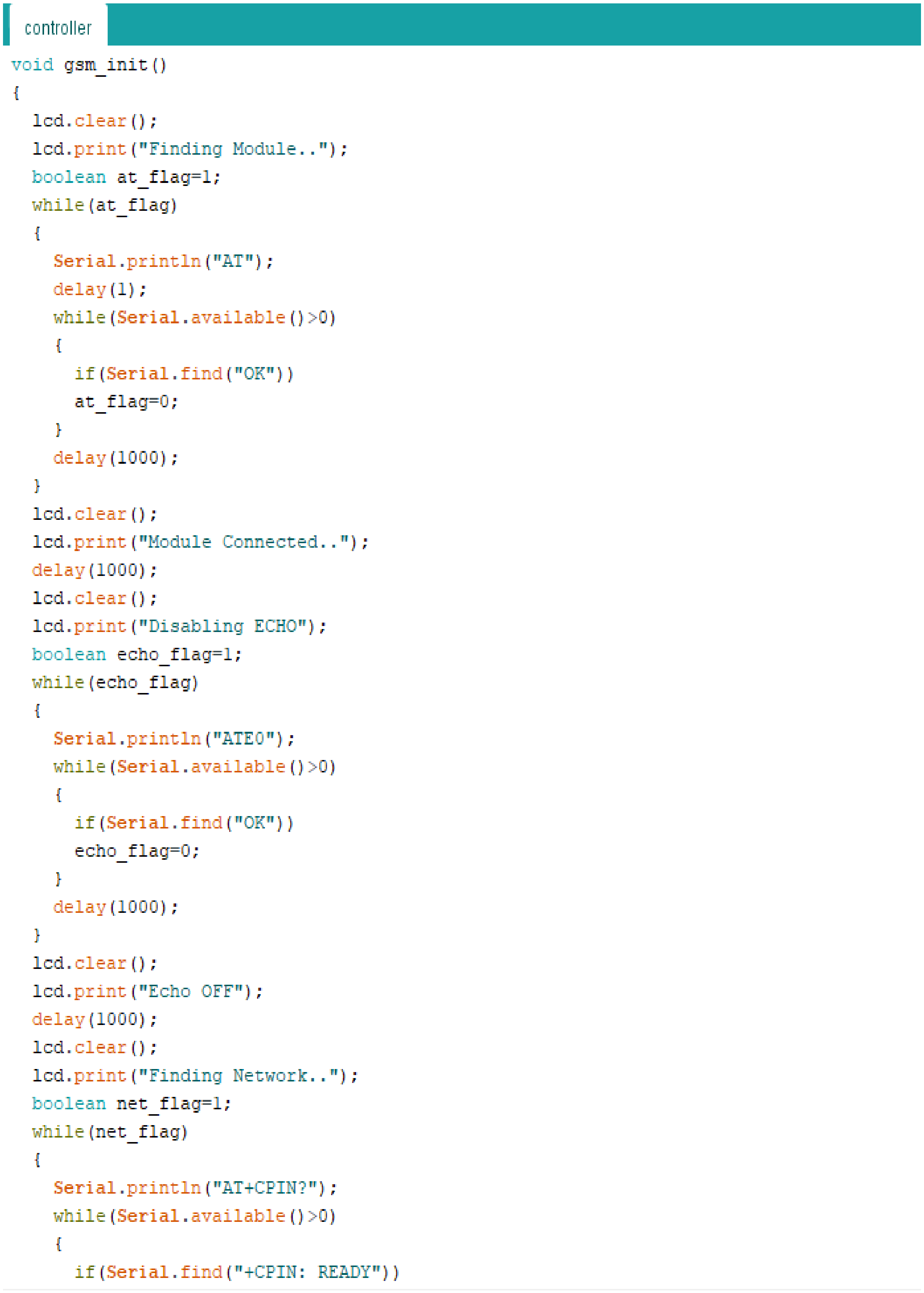
Figure 5. 9 Design of controller

5.5 Prototype

Source code to initialize controller



```
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
#include<LiquidCrystal.h>
LiquidCrystal lcd(13, 12, 11, 10, 9, 8);
static const int RXPin = 4, TXPin = 3;
static const uint32_t GPSBaud = 9600;
// The TinyGPS++ object
TinyGPSPlus gps;
int temp=0,i;
// The serial connection to the GPS device
SoftwareSerial ss(RXPin, TXPin);
String stringVal = "";
void setup(){
  Serial.begin(9600);
  ss.begin(GPSBaud);
  lcd.begin(16,2);
  pinMode(13,OUTPUT);
  digitalWrite(13,LOW);
  lcd.print("Vehicle Tracking");
  lcd.setCursor(0,1);
  lcd.print("    System      ");
  delay(2000);
  gsm_init();
  lcd.clear();
  Serial.println("AT+CNMI=2,2,0,0,0");
  lcd.print("GPS Initializing");
  lcd.setCursor(0,1);
  lcd.print("  No GPS Range  ");
  delay(2000);
  lcd.clear();
  lcd.print("GPS Range Found");
  lcd.setCursor(0,1);
  lcd.print("GPS is Ready");
```



```
void gsm_init()
{
  lcd.clear();
  lcd.print("Finding Module..");
  boolean at_flag=1;
  while(at_flag)
  {
    Serial.println("AT");
    delay(1);
    while(Serial.available()>0)
    {
      if(Serial.find("OK"))
      at_flag=0;
    }
    delay(1000);
  }
  lcd.clear();
  lcd.print("Module Connected..");
  delay(1000);
  lcd.clear();
  lcd.print("Disabling ECHO");
  boolean echo_flag=1;
  while(echo_flag)
  {
    Serial.println("ATE0");
    while(Serial.available()>0)
    {
      if(Serial.find("OK"))
      echo_flag=0;
    }
    delay(1000);
  }
  lcd.clear();
  lcd.print("Echo OFF");
  delay(1000);
  lcd.clear();
  lcd.print("Finding Network..");
  boolean net_flag=1;
  while(net_flag)
  {
    Serial.println("AT+CPIN?");
    while(Serial.available()>0)
    {
      if(Serial.find("+CPIN: READY"))
```

5.6 Deployment

Django, being a web framework, needs a web server in order to operate. And since most web server doesn't natively speak python, we need an interface to make that communication happen. Django currently support two interfaces AND ASGI but we use WSGI interfaces. Due to the fact that ASGI server is newer and therefore tested less, have less features and fewer number of communities, we use WSGI interfaces for our project deployment. Technology used for deployments is:

- **NGINX**: - is a well-known high-performances Http server and reverse proxy. NGINX provides an abundance of features, such as load balancing, caching's/TLS offloading, log rotation, and so on.
- **HEROKU**: - is cloud application platform, it is basically a platform-as-a-Service (paaS). They support several programming languages, including python. They also offer a free plan, which is quite limited, but it is great standard and to host demos of Django applications.

Component name	Implementation detail
Authentication Controller	Implemented using python code on Django framework to perform Authentication.it is required when the user wants to login.it checks the validity of user login inputs
Page controller	This is a router class which will direct the user to each intended pages according to the user actions. It will be implemented using python code on Django framework.
User view	Is the view that interact with user.
API controller	Is a controller which uses some parameters and interact with other API based systems to fetch data. this function is controlled by the Nginx web server.
Reservation controller	Is a controller which uses user reservation input and after validation uses the API controller get the reservation token from services provider system and end booking process.
Load Balancer	This is the component of the Nginx web server that which checks which server is able to handle user requests.
User search filtering controller	Controls the filtering process for the searched services provider using some filtering attributes.

Table 5. 4 Deployment

CHAPTER SIX

SYSTEM TESTING

6.1 Introduction

System testing is defined as testing of a complete and fully integrated software product. It is performed in the context of a System Requirement Specification (SRS) and/or a Functional Requirement Specifications (FRS). This chapter will cover what are the testing methods used by our team and what are the results obtained from the test are.

6.2 Scope

The testing plan will focus on performing tests for existing modules and testing the integrated modules. This means, first, all the component of the EVSC system will be tested one by one and then will be integrated together and tested. The task of this section documentation is:

- Test the functionality of the features,
- Test the flawless working of module integrations
- Ensure proper data creation at the database

This testing is done for:

System name	EVSC system
System Version	1.0.0

Table 6. 1 Test

6.3 Resources

For the implementation of testing, we will use SRS document of the project.

6.4 Schedule

Test activities will be carried out on each module according to the following time table.

T01(Test 1)

Test type: Unit Testing
Test description: All components of the system will be tested individually.
Test by:

Amhir Edris: Tests EVSC mobile app that traffic polices use

Amanuel Girma: Hardware components functionality.

Meheratu Tamiru: Tests EVSC web part

Test duration: August 2, 2021- August 10, 2021

T02(Test 2)

Test type: Integration Testing
Test description: Testing whether the modules which were working fine individually, does or does not have issues when integrated together.

Test by: Mekuanent Molla and Lencha Fikiru

Test duration: August 11, 2021- August 16, 2021

T03(Test 3)

Test type: System testing
Test description: testing of a complete and fully integrated EVSC system.

Test by: Meheratu Tamiru and Lencha Fikiru

Test duration: August 17, 2021- August 22, 2021

6.5 Features to be tested and not to be tested

6.5.1 Features to be tested

Are the features of the system or sub-system that will be tested. Identifies the systems operations, scenarios, and functionality that are to be tested in each system and sub-system as these are what deliver value to the issue being solved.

no	Test targets	Feature to be tested	Tested by
1	Functional requirement	General functionality of the system	Lencha Fikiru and Mekuanent Molla
2	Code	Each code component of the software	Amhir Edris and Amanuel Girma
3	Hardware status	Testing all hardware part functionality	Meheratu Tamiru

Table 6. 2 Features to be tested

6.5.2 Features not to be tested

Defines features of the system or sub-system that will NOT be tested. This is necessary so as to avoid later confusion when stakeholders thought something would be tested, but was not. Non-functional requirements are not tested

Test Target	**Feature to be tested**	**Tested by**
Non-Functional requirements	Security, Response time	Mekuanent Molla and Lencha Fikiru

Table 6. 3 Features not to be tested

6.6 Pass/fail criteria

It specifies the criteria that will be used to determine whether each test item has passed or failed testing.

Functional requirement	**Expected result**	**Actual outcome**	**Pass/fail**
Login	Pass	Pass	Pass
Send alert	Pass	Pass	Pass
Send notification	Pass	Pass	Pass
Receive alert	Pass	Pass	Pass
View vehicle detail	Pass	Pass	Pass
View traffic police detail	Pass	Pass	Pass

Table 6. 4 Pass/fail criteria

Approach

In our project we have used proactive approach. The test design process is started as early as possible in order to find and fix the defects before the build is created.

With this testing approach, all the specifications were ready for a prototype, and a plan was already built to be shown; the tester started writing his or her code and saw if he or she could obtain the same results that the specs mentioned. This way, the specifications were tested on each prototype, and continuous testing was applied. This also helped to minimize the testing that would have to be implemented at the end of the software lifecycle. In this process, identified aspects of the software were tested.

6.7 Test case specification

Test specifications of functional requirements used to write the test cases along with the test case numbers for each test case and a short description of the test cases are shown in the table below.

Test case Identifier	**Test case name**	**Test case short description**
TC01	Login	To test authentication interface for the user.
TC02	Over speed detection	To test whether the controller detects over speeding.
TC03	Vehicle location	To test sending of notification to the nearest traffic police.
TC04	Nearest traffic police identification	To test the system displays the required details.

TC05	Vehicle detail	In order to test vehicle detail display
TC06	Receiving alert	To test communication between controller and EVSC system
TC07	Assign traffic police location	To test population of traffic police locations
TC08	Sending report	To test report submission to EVSC system
TC09	Registration	To test adding vehicle and traffic police data to the database
TC10	Change password	To test whether the system allows password change or not.

Table 6. 5 Test case specification

The following list includes the steps that should be taken by the user, the conditions that should be met for the successful execution of the test case, and the end result that should be met for the test cases to pass.

1. TC01: Login
    - Input: Username and Password.
    - Output: valid destination Activity.
    - Valid range: Username=Alphanumeric, password=capital, small letter, number and characters.
    - End message result:
        - i. If (User = = valid), a success message is displayed and redirected to home activity.
          ii. If (User! = valid), system prompts error message on the login activity.
2. TC02: Over speed detection
    - Precondition: speed is detected by the controller.
    - Input: Vehicle speed and Speed limit
    - Output: Over speeding or not over speeding.
    - Valid range: Below the speed limit.
    - End result message:
      i. If (Speed > speed limit), send alert message to EVSC system
      ii. If (Speed < = speed limit), continue
3. TC03: Vehicle location
    - Preconditions: GPS actively detecting location
    - Input: Longitude and latitude
    - Output: Vehicle location
4. TC04: Nearest traffic police identification

- Assumptions: Traffic polices are at their assigned position
- Input: Location where traffic polices are assigned and vehicle location
- Output: Nearest traffic police location

5. TC05: View vehicle detail
   - Assumption: Logged in and on the homepage of the EVSC web or app
   - Test steps:
     1. click on the view vehicle detail
     2. select one from the existing records or search for with plate number of the vehicle
     3. view vehicle detail
   - Expected result: Vehicle histories being displayed from database
6. TC06: EVSC system receiving alert from Controller
   - Description: Sending alert and receiving alert are real-time processes. They cannot be tested directly. To test whether an alert is received or not, database data has to be checked.
   - Precondition: Controller sending alert
   - Input: alert from controller
   - Output: Notification sent to traffic police and displayed to admin
7. TC07: Assign traffic police location
   - Precondition: Assigning admin has logged in.
   - Input: Traffic police identity, longitude and latitude
   - Output: Assigned traffic police data with location it was assigned to.
8. TC08: Sending report
   - Assumption: Traffic police has logged into EVSC app.
   - Precondition: Traffic police has taken an action.
   - Expected output: Report sent to EVSC system
9. TC09: Registration
   - Precondition: System admin on EVSC homepage
   - Input: Vehicles or traffic polices attributes
   - Expected output: New data entry added to database table accordingly
10. TC10: Change password
    - Assumption: User has logged in.
    - Input: Old password and new password
    - Expected output: password changed
    - Produced output:
    1. when correct old password is inserted with allowed combination of new password, password is changed
    2. When wrong old password or wrong combination of new passwords are inserted, error message is displayed.

6.8 Estimated risk and contingency plan

The following are the risks identified to be faced during the development of the system and with contingency plan that developed to solve or minimize the risks.

Estimated risk	**Contingency plan**
Lack of time to develop the system fully	Use time wisely
Lack of knowledge on designing	Allocate more time on design works
Communication between hardware components	Allocate more time on implementing the communication between hardware components.
Lack of different hardware tools	Make inquire to ECE department or buy
Lack of knowledge on programming language.	Allocate more time on understanding that programming language.

Table 6. 6 Estimated risk and contingency plan

CHAPTER 7

CONCLUSION AND RECOMMENDATION

7.1 Conclusion

The application of GPS, GSM and GPRS in over-speeding detection and notifying is proven to be the most effective solution compared to other existing other methods like safety speed gun, cameras, human inspection and speed governors.

This project resulted in the development of A Real-time Vehicle overspeed detection and notifying system. The system was built based on several integrated modules. The system we developed has both hardware and software components integrated together to control vehicle speed limits and notifies overspeed vehicle cases to the control system and then to the traffic polices on duties in real-time. The system also provides reporting actions taken by traffic polices to the control system. It uses best rout finding algorithms to send an alarm to the nearest traffic polices on duties.

All the violations and actions taken are stored on the database which can be used for later analysis on speed violation patterns.

While developing this system we have gained more knowledge on how to design and assemble hardware modules, programming languages like c/c++ (for the Arduino system parts), Android. over all we can say that we have implemented what we gain from theoretical part of knowledge to practical work.

7.2 Recommendations

For the betterment of this system, populating each and every Ethiopian road speed limit to the google map will make the system more accurate and efficient. Configuring more segments on the controller part can also enhance the system. Manufacturing all the hardware devices configured together in one pack prevents this device form temper.

Reference

1. [1] “Automated Over Speeding Detection and Reporting System” vol. 4, no. 4, pp. 613–619, 2015.

2. [2] “Design & Analysis of Vehicle Speed Control Unit Using RF Technology” vol. 2, Issue 8, August 2015.

3. [3] “M. Rouse, “What is GSM (Global System for Mobile communication)? - Definition from WhatIs.com.” [Online]. Available:”

4. Cao J., Wei J., & Qin Y. (2013). Research and Application of the Four-tier Architecture. Guiyang, China: Atlantis Press

5. Dr. Kamal Jain and Rahul Goel “GPS Based Low-Cost Intelligent Vehicle Tracking System (IVTS)”, 2012 International Conference on Traffic and Transportation Engineering (ICTTE 2012), IPCSIT vol. 26

6. Global status report on road safety: time for action. Geneva, World Health Organization, 2009 (www.who.int/violence_injury_prevention/road_safety_status/2009).

7. “GSM @ searchmobilecomputing.techtarget.com.”

8. http://searchmobilecomputing.techtarget.com/definition/GSM.

9. “Raspberry Pi” [Online]. Available: https://elinux.org/RPi_Hub. [Accessed: 01-Sep- 2017].

APPENDEXES

Appendix A: Controller Code sample

Description: This code sample below sends speed to the server

```
#include <TinyGPS++.h>

#include <WiFi.h>

#include <HTTPClient.h>

const char* ssid     = "AD";

const char* password = "password";

const char* SERVER_NAME

unsigned long previousMillis = 0;

long interval = 20000;

#define rxPin 4

#define txPin 2

HardwareSerial neogps(1);

TinyGPSPlus gps;

void setup() {

  Serial.begin(115200);

  Serial.println("esp32 serial initialize");

  neogps.begin(9600, SERIAL_8N1, rxPin, txPin);

  Serial.println("neogps serial initialize");

  delay(1000);

  WiFi.begin(ssid, password);

  while(WiFi.status() != WL_CONNECTED) {

    delay(500);

  }

}

void loop() {
  //Check WiFi connection status
  if(WiFi.status()== WL_CONNECTED){
    unsigned long currentMillis = millis();
    if(currentMillis - previousMillis > interval) {
       previousMillis = currentMillis;
       //Send an HTTP POST request every 30 seconds
       sendGpsToServer();
    }
  }
  else {
    Serial.println("WiFi Disconnected");
  }
  delay(1000);
}
int sendGpsToServer()
{
  boolean newData = false;
  for (unsigned long start = millis(); millis() - start < 2000;){
    while(neogps.available()){
      if(gps.encode(neogps.read())){
        if(gps.location.isValid() == 1){
          newData = true;
          break;
        }
      }
    }
```

```
 }
 //If newData is true
 if(true){
   newData = false;
   String latitude, longitude, plate_number='AA121221';
   unsigned long date, time, speed;
   latitude = String(gps.location.lat(), 6); // Latitude in degrees (double)
   longitude = String(gps.location.lng(), 6); // Longitude in degrees (double)
   date = gps.date.value(); // Raw date in DDMMYY format (u32)
   time = gps.time.value(); // Raw time in HHMMSSCC format (u32)
   speed = gps.speed.kmph();
   HTTPClient http;
   http.begin(SERVER_NAME);
   String gps_data;
   gps_data += "&lat="+latitude;
   gps_data += "&lng="+longitude;
   gps_data += "&pno="+plate_number;
   gps_data += "&spe="+speed;
   gps_data += "&lng="+date;
   gps_data += "&lng="+time;
   Serial.print("gps_data: ");
   Serial.println(gps_data);
   int httpResponseCode = http.POST(gps_data);
   String httpResponseString = http.getString();
      http.end();
 }
}
```

Appendix B: EVSC system code Sample
Description: Code below helps to find shortest path

```
from django.db.models.signals import post_save
from django.contrib.auth.models import User
from .models import TrafficPolice,SystemAdmin,Records,Notification,Report, TrafficPoliceLocation
from django.dispatch import receiver
from pyfcm import FCMNotification

import math
@receiver(post_save,sender=User)
def register_user(sender, instance, created, **kwargs):
    print("created: ",created)
    if created:
        if instance.is_staff !=True:
            TrafficPolice.objects.get_or_create(user=instance)

        else:
            SystemAdmin.objects.get_or_create(user=instance)
# sending Notification instantaneously as soon as new record is registered

@receiver(post_save,sender = Records)
def send_notification(sender,instance,created,**kwargs):
    print("created :", created)

    if created:

        traffic_police_location_list = []
        all_traffic_police = TrafficPolice.objects.all()
        all_traffic_police_count =TrafficPolice.objects.all().count()
      traffic_police_location_count = TrafficPoliceLocation.objects.all().count()

        # append all traffic police location to the list
        for traffic_police in all_traffic_police:
            if traffic_police.location == None:
                continue
            traffic_police_location_list.append({"latitude": traffic_police.location.latitude, "longitude": traffic_poli
ce.location.longitude,"obj": traffic_police})



        # cheack weather the traffic police location exists or not
        if len(traffic_police_location_list) > 0:
            # calculate distance between record and all traffic police
            list_of_distance = []
            record_latitude = instance.latitude
            record_longitude = instance.longitude
            radius = 6371
            for traffic_police_pair in traffic_police_location_list:
                traffic_latitude = traffic_police_pair["latitude"]
                traffic_longitude = traffic_police_pair["longitude"]
                traffic_obj = traffic_police_pair["obj"]
                # create list of neare by traffic police

                dlat = math.radians(traffic_latitude-record_latitude)
                dlong = math.radians(traffic_longitude-record_longitude)
                a = math.sin(dlat/2) * math.sin(dlat/2) + math.cos(math.radians(record_latitude)) \
                * math.cos(math.radians(traffic_latitude)) * math.sin(dlong/2) * math.sin(dlong/2)
                c = 2 * math.atan2(math.sqrt(a), math.sqrt(1-a))
                d = radius * c

                list_of_distance.append((d, traffic_obj))

            # minimum distance
            minimum_distance_pair = min(list_of_distance)
            nearby_traffic_police = minimum_distance_pair[1]
            print(minimum_distance_pair[0], nearby_traffic_police)
            # minimum_distance = index(min(list_of_distance))

            print(nearby_traffic_police.fcm_token)

        notification = Notification.objects.create(recipient = nearby_traffic_police, records = instance, content = "
message_body")
        notification.save()

        print(instance.vehicle.vehicle_plate)
        # create instance of FCMNotification Class by providing API Key
        push_service = FCMNotification(api_key = 'AAAAbG5wAg0:APA91bH60qfGn4rg2B-
2bSWicLWShygvmNrlrSX0LM9VzM9Srqcxvo3XIX9ODSrk92Zhuk4kPQ10V5DCRVVzDXN7koQSSP7S8
aQhtRZQEULS10nL57k_Ote3AQzcolVRcuCnV8NgcGdw')
        #retrieve registration Id of Traffic Police from The database
        registration_id = nearby_traffic_police.fcm_token
        print(registration_id)
        message_title = 'Notification test'
        message_body = "Hi Aman, We made it bro!"
        # sending Notification to Single Devices
        result = push_service.notify_single_device(registration_id=registration_id, message_title=message_title,
message_body=message_body)
        print(result)
```

Appendix C: sample code of EVSC Android app

Notification

To receive notification

Description: Used to Receive Notification from firebase and Display it to the Mobile Device

```
package com.amanuel.evscsystem.fcm

import android.util.Log
import androidx.navigation.NavDeepLinkBuilder
import androidx.work.OneTimeWorkRequest
import androidx.work.WorkManager
import com.amanuel.evscsystem.R
import com.amanuel.evscsystem.notification.NotificationHelper
import com.google.firebase.messaging.FirebaseMessagingService
import com.google.firebase.messaging.RemoteMessage

class FirebaseCloudMessagingService : FirebaseMessagingService() {

    companion object {
        private const val TAG = "MyFirebaseMsgService" // used for logging data
        private const val TAG_FCM_MESSAGE = "MessageFromFCM"
        var notificaitonId: Int = 0
    }


    // [START receive_message]
    override fun onMessageReceived(remoteMessage: RemoteMessage) {
        super.onMessageReceived(remoteMessage)
        // TODO(developer): Handle FCM messages here.

        // Not getting messages here? See why this may be: https://goo.gl/39bRNJ
        Log.d(TAG, "From: ${remoteMessage.from}")

        // Check if message contains a data payload.
        if (remoteMessage.data.isNotEmpty()) {
            Log.d(TAG, "Message data payload: ${remoteMessage.data}")

            if (/* Check if data needs to be processed by long running job */ true) {
                // For long-running tasks (10 seconds or more) use WorkManager.
                scheduleJob()
            } else {
                // Handle message within 10 seconds
                handleNow()
            }
        }

        // Check if message contains a notification payload.
        remoteMessage.notification?.let {
            Log.d(TAG, "Message Notification Body: ${it.body}")

        }

        // Also if you intend on generating your own notifications as a result of a received FCM
        // message, here is where that should be initiated. See sendNotification method below.
        if (remoteMessage.data.isNotEmpty()) {
            // create the deep link using the pending intent
            // optionally we can pass in an argument using setArgument method
            val pendingIntent = NavDeepLinkBuilder(this)
                .setGraph(R.navigation.nav_graph_main)
//                .setDestination(R.id.notificationsDetailFragment)
                .setDestination(R.id.notificationsFragment)
                .createPendingIntent()


            NotificationHelper.postNotification(
                notificaitonId,
                remoteMessage.data["title"].toString(),
                remoteMessage.data["body"].toString(),
                this,
                pendingIntent
            )
        }
    }
    // [END receive_message]

    // [START on_new_token]
    override fun onNewToken(token: String) {
        Log.d(TAG, "Refreshed token: $token")
    }
    // [END on_new_token]


    /**
     * Schedule async work using WorkManager.
     */
    private fun scheduleJob() {
        // [START dispatch_job]
        val work = OneTimeWorkRequest.Builder(MyWorker::class.java).build()
        WorkManager.getInstance().beginWith(work).enqueue()
        // [END dispatch_job]
    }

    /**
     * Handle time allotted to BroadcastReceivers.
     */
    private fun handleNow() {
        Log.d(TAG, "Short lived task is done.")
    }

}
```

Notification channel creator

```
package com.amanuel.evscsystem.notification

import com.amanuel.evscsystem.R
import android.app.Activity
import android.app.NotificationChannel
import android.app.NotificationManager
import android.app.PendingIntent
import android.content.Context
import android.graphics.Color
import android.os.Build
import androidx.appcompat.app.AppCompatActivity
import androidx.core.app.NotificationCompat
import androidx.core.app.NotificationManagerCompat

/**
 * created by: Amanuel Girma
 *
 */



/**
 * Utility class for posting notifications.
 * This class creates the notification channel (as necessary) and posts to it when requested.
 */
object NotificationHelper {

    private const val channelId = "Default"

    fun init(activity: Activity) {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            val notificationManager =
                activity.getSystemService(AppCompatActivity.NOTIFICATION_SERVICE) as
NotificationManager
            val existingChannel = notificationManager.getNotificationChannel(channelId)
            if (existingChannel == null) {
                // Create the NotificationChannel
                val name = activity.getString(R.string.defaultChannel)
                val importance = NotificationManager.IMPORTANCE_DEFAULT
                val mChannel = NotificationChannel(channelId, name, importance)
                mChannel.description = activity.getString(R.string.notificationDescription)
                notificationManager.createNotificationChannel(mChannel)
            }
        }
    }

    fun postNotification(
        id: Int,
        title: String,
        body: String,
        context: Context,
        intent: PendingIntent
    ) {
        val builder = NotificationCompat.Builder(context, channelId)
        builder.setContentTitle(title).setSmallIcon(R.drawable.ic_notifications)
        builder.color = Color.parseColor("#002171") // secondaryDarkColor
        val notification = builder.setContentText(body)
            .setPriority(NotificationCompat.PRIORITY_MAX)
            .setContentIntent(intent)
            .setAutoCancel(true)
            .build()
        val notificationManager = NotificationManagerCompat.from(context)
        // Remove prior notifications; only allow one at a time to edit the latest item
//        notificationManager.cancelAll()
        notificationManager.notify(id, notification)
    }
}
```

C) **Notification Search Queries** on Android's Room Database

```
package com.amanuel.evscsystem.data.db

import androidx.room.Dao
import androidx.room.Insert
import androidx.room.OnConflictStrategy
import androidx.room.Query
import com.amanuel.evscsystem.data.db.models.Notification
import com.amanuel.evscsystem.ui.notification.SortOrder
import kotlinx.coroutines.flow.Flow

@Dao
interface NotificationDao {


    fun getNotifications(searchQuery: String = "", sortOrder: SortOrder): Flow<List<Notification>> =
        when(sortOrder){
            SortOrder.BY_DATE -> getNotificationsSortedByDateCreated(searchQuery)
            SortOrder.BY_PLATE_NUMBER -> getNotificationsSortedByName(searchQuery)
        }

    @Query("SELECT * FROM Notification WHERE plate_number LIKE '%' || :searchQuery || '%'
ORDER BY createdAt")
    fun getNotificationsSortedByDateCreated(searchQuery: String = ""): Flow<List<Notification>>

    @Query("SELECT * FROM Notification WHERE plate_number LIKE '%' || :searchQuery || '%'
ORDER BY plate_number")
    fun getNotificationsSortedByName(searchQuery: String = ""): Flow<List<Notification>> // name here is
similar to plateNumber

//    @Query("SELECT * FROM Notification")
//    fun getNotifications(): Flow<List<Notification>>

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insert(notification: Notification)

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertAll(notificationList: List<Notification>)

    @Query("DELETE FROM Notification")
    suspend fun deleteAll()
}
```

Notification Model Format

Description: Notification model format that will be Received from the Server and Stored locally on android's Room Database

```
import android.os.Parcelable
import androidx.room.Entity
import androidx.room.PrimaryKey
import kotlinx.parcelize.Parcelize
import java.text.DateFormat

// Note: the Parcelable Interface makes it easy to send
// the object between different fragments
@Entity
@Parcelize
data class Notification(
    @PrimaryKey(autoGenerate = true) val id: Int,
    val notification_id: Int,
    val record_id: Int,
    val latitude: Double,
    val longtude: Double,
    val plate_number: String,
    val vehicle_speed: Int,
    val Location: String?, // must be removed
    val createdAt: Long? = System.currentTimeMillis(),
) : Parcelable {
    val createFormattedDate: String
        get() = DateFormat.getDateTimeInstance().format(createdAt)
}
```

1. Retrofits Api Interface for getting Notification

```
package com.amanuel.evscsystem.data.network

import com.amanuel.evscsystem.data.db.models.Notification
import retrofit2.Response
import retrofit2.http.GET
import retrofit2.http.Headers

interface NotificationApi {

    @Headers("Accept: application/json")
    @GET("notification/?format=json")
    suspend fun getNotifications(): List<Notification>
}
```

Report

A Report model format

```
package com.amanuel.evscsystem.data.db.models

import com.google.gson.annotations.SerializedName

data class Report(
    @SerializedName("is_drunk") val isDrunk: Boolean,
    @SerializedName("is_using_cell_phone") val isUsingCelPhone: Boolean,
    @SerializedName("is_not_having_license") val isNotHavingLicense: Boolean,
    @SerializedName("is_chewing_chat") val isChewingChat: Boolean,
    @SerializedName("other_violation") val otherViolation: String,
    @SerializedName("description") val description: String,

    // measures taken
    @SerializedName("payment_amount") val paymentAmount: Double,
    @SerializedName("short_summary") val shortSummary: String,
)
```

Reporting Fragment

```
import android.os.Bundle
import android.view.View
import android.widget.Toast
import androidx.appcompat.app.AlertDialog
import androidx.fragment.app.Fragment
import androidx.fragment.app.viewModels
import androidx.navigation.fragment.findNavController
import com.amanuel.evscsystem.R
import com.amanuel.evscsystem.data.db.models.Report
import com.amanuel.evscsystem.databinding.FragmentReportBinding
import com.amanuel.evscsystem.utilities.EVSCDialogMsg
import com.amanuel.evscsystem.utilities.showErrorSnackBar
import dagger.hilt.android.AndroidEntryPoint
import kotlin.properties.Delegates

@AndroidEntryPoint
class ReportFragment : Fragment(R.layout.fragment_report) {

    private lateinit var binding: FragmentReportBinding

    private val viewModel: ReportViewModel by viewModels()

    private var recordId by Delegates.notNull<Int>()
    private var plateNumber by Delegates.notNull<String>()

    private lateinit var arrayViolations: Array<String>
    private lateinit var arrayChecked: BooleanArray


    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        binding = FragmentReportBinding.bind(view)

        val args = ReportFragmentArgs.fromBundle(requireArguments())
        recordId = args.recordId
        plateNumber = args.plateNumber

        Toast.makeText(requireActivity(), "Record: $recordId", Toast.LENGTH_SHORT).show()

        arrayViolations = resources.getStringArray(R.array.other_violations_array)
        arrayChecked = BooleanArray(arrayViolations.size)
    binding.otherViolationsTextView.setOnClickListener {
        showDialog()
    }


    // handle the click event of the report button
    binding.reportButton.setOnClickListener {
        report()
    }

}

private fun report() {
    val isDrunk: Boolean = binding.isDrunkCheckBox.isChecked
    val isUsingCellPhone: Boolean = binding.isUsingCellPhoneCheckBox.isChecked
    val isNotHavingLicense: Boolean = binding.isNotHavingLicenceCheckBox.isChecked
    val isChewingChat: Boolean = binding.isChewingChatCheckBox.isChecked
    val otherViolations: String = binding.otherViolationsTextView.text.toString()
    val description: String = binding.descriptionEditTextTextMultiLine.text.toString()

    var paymentAmount: Double
    if (binding.paymentAmountTextField.text.toString().isNotEmpty()) {
        paymentAmount = binding.paymentAmountTextField.text.toString().toDouble()
    } else {
        paymentAmount = 0.0
    }

    val shortSummary: String = binding.shortSummaryEditTextTextMultiLine.text.toString()

    val report = Report(
        isDrunk,
        isUsingCellPhone,
        isNotHavingLicense,
        isChewingChat,
        otherViolations,
        description,
        paymentAmount,
        shortSummary
    )

    view?.let {
        viewModel.report(it, recordId, report) { report ->
            if (report != null) {
                // it = newly added user parsed as response
                // it?.id = newly added user ID
                Toast.makeText(requireContext(), "reporting is success!", Toast.LENGTH_SHORT).show()

                EVSCDialogMsg.showSuccessAlert(
                    requireContext(),
                    "Reporting Success!!!",
                    "You have successfully written a report for vehicle with plate number: $plateNumber"
                ) { dialog, which ->
                    findNavController().navigate(R.id.action_reportFragment_to_notificationsFragment)
                }
                } else {
                    view?.showErrorSnackBar("Reporting Failure!!!")
                }
            }
        }
    }


    // Method to show an alert dialog with multiple choice list items
    private fun showDialog() {
        // Late initialize an alert dialog object
        lateinit var dialog: AlertDialog

        // Initialize a new instance of alert dialog builder object
        val builder = AlertDialog.Builder(requireContext())

        // Set a title for alert dialog
        builder.setTitle("Choose Other Violations.")

        // Define multiple choice items for alert dialog
        builder.setMultiChoiceItems(arrayViolations, arrayChecked) { dialog, which, isChecked ->
            // Update the clicked item checked status
            arrayChecked[which] = isChecked

            // Get the clicked item
            val violation = arrayViolations[which]

            // Display the clicked item text
            Toast.makeText(requireContext(), "$violation clicked.", Toast.LENGTH_SHORT).show()
        }

        // Set the positive/yes button click listener
        builder.setPositiveButton("OK") { _, _ ->
            // Do something when click positive button
            binding.otherViolationsTextView.text = ""
            for (i in 0 until arrayViolations.size) {
                val checked = arrayChecked[i]
                if (checked) {
                    binding.otherViolationsTextView.text =
                        "${binding.otherViolationsTextView.text}  ${arrayViolations[i]} \n"
                } else {
                    arrayChecked[i] = false
                }
            }
        }
        // Initialize the AlertDialog using builder object
        dialog = builder.create()
        // Finally, display the alert dialog
        dialog.show()
    }

}
```