VEHICLE CRASH LOCATION DETECTION AND ALERTING SYSTEM USING ANDROID SMART PHONE AND GPS TECHNOLOGIES

Nowadays varying degrees of road crashes have been recorded from both developed and developing nations of the world. Crashes especially those that involve loss of lives around the world and in particular in developing countries is alarming. The existing solution is in-vehicle accident detection system, which is expensive. In this research work, a less expensive and system that can be widely used to reduce the time to report and respond to vehicle crash was developed. The system consists of two mobile-phone-based applications, the User or victim's App and the agency App. The victim App uses the in-phone accelerometer and General Packet Service (GPS) to detect a crash and its location coordinates respectively, and then a notification is sent to a concerned agency. A maximum of three concerned agencies location coordinates with their respective phone numbers were stored on a firebase (cloud database) account for mapping to the closest crash site and then send an SMS with the location coordinates and the Google map image of the crash site in real-time . The agency App is used to register an agency on the firebase. To be able to view the Google map images, each of the concerned agencies had a smart phone. The two Apps were developed and implemented using Android Studio Integrated Development Environment (IDE), with Java programming language for the events and Extended Mark-Up Language (XML) for the interface design. The implementation was strictly on Android smart phones with minimum Application Programming Interface (API) level 16. The system validation proves high sensitivity of 93%, accuracy of 91%, precision of 92% and specificity of 87% which shows that the system is efficient and effective.

CHAPTER ONE

1.0 INTRODUCTION

1.1 Background of the Study

Accidents caused by vehicular traffics results in a lot of human injuries and death. Delay in dispatching emergency aids to accident location is one of the most important indicators for survival rate. A vehicle equipped with a system that can automatically detects accident and quickly alert first responders is one of the ways used to eliminate such delays, (White *et al*., 2011)**.** The majority of vehicles using our roads do not have this automatic detection system and it is expensive to retrofit for older vehicles.

Furthermore, traffic is on the rise as the demand for vehicles is getting higher, so the means of transportation needs to be improved upon. Accident victims need to get help on time to reduce fatality rate. Figure 1.1 shows a crashed vehicle with an in-built accident detection system. The crash activates auto dialer inside the car and sends GPS location and the vehicle registration number to a first responder, an insurance company if the vehicle has insurance and to family members.

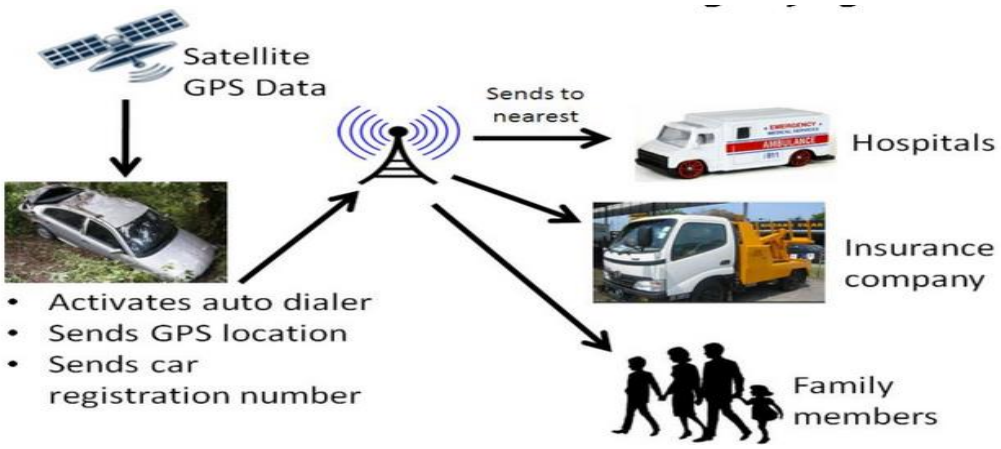


Fig.1.1 Automatic Accident Location Detection System (AALDS)
(Goh *et al.*, 2014)

However, various accident detection systems have been proposed with different methodologies. One of such methods is the use of the Arduino micro controller with other components like the GPS receiver and GSM/GPRS components in other to detect and send location of an accident for quick response of first responders. Real-time control of vehicle speed automatically could be very difficult to realize, instead of controlling the vehicle speed automatically, a system that detects accidents and also keep the driver of a vehicle on alert about the speed limit was proposed. The system can detects when a driver of a vehicle exceeds the speed limits and ignores the warning. Using the GSM technology incorporated in the system design, an SMS will be sent to a patrol corps with the location of the speeding vehicle, (Narendar and Teja, 2013).

In a world that is becoming more motorized, impact sensors in conjunction with GPS offers a timely, objective and potentially acceptable method of detecting and reporting accidents, (Njuguna, 2012).

In addition, smart phones are cheap and easy to purchase in comparison with other traffic accident prediction system, this makes them a very good alternative to use. Moreover, even if a car is not fitted with an accident detection and notification system, smart phones generally travel with their owners to detect accidents, (White *et al*., 2011).

1.2 Statement of the Problem

When a vehicular accident occurs, it is usually the good Samaritans that report such accidents to the emergency responders. The moment between when an accident happens to the time where such emergency centers are notified is a key indicator to saving lives. Waiting for good Samaritans to report accident will lead to wasting precious time which the victims may not have. Having an automatic notification system that can notify the emergency center that is closest to the accident site will go a long way in reducing the level of causalities. In addition, the various in-built car crash detection and alerting systems are mostly available in luxury cars. These cars are mostly saloon and have maximum sitting capacity from 5-9 people commuting per vehicle. The majority of vehicles on our roads don't have this facility. Android smart phones are one of the cheapest smart phones available and it is believed that at least one in every five people possesses it. Utilizing these facilities in android smart phone to develop a system that can detect accidents will reduce the fatality rate of accidents.

1.3 Aim and Objectives of the Research Study

The aim of this research work is to develop a vehicle crash location detection and alerting system using android smart phone and GPS technologies.

The specific objectives of the study were:

i. To set an acceleration threshold for a smart phone that can detect crash using the smart phone accelerometer and GPS technologies.
ii. To develop an algorithm that maps an accident location to an emergency center closest to the accident location and send notification to that center.
iii. To evaluate the performance of the developed system using accuracy, precision, speed, sensitivity and cellular network type as the performance metrics.

1.4 The Significance of the Study

The research work will eliminate the delay in notifying an accident to first responders such as the Nigerian Road Safety Cooperation and Red Cross. Automatic notification using the android smart phone and the GPS technologies will provide timely and accurate information to these organizations and eliminate delay in time between the occurrence of an accident to when such accident is reported. This will go a long way in saving more lives of accident victims because the required aid and attention will be brought to the accident site immediately.

1.5 Scope of the Study

In the proposed system, only android smart phones were chose and used for the system test. Even though there are other smart phones, the availability and the ease of use of the android technology prompted the decision. A remote controlled toy car was used to simulate the vehicle that will be involved in an accident; this is because using a real vehicle will be expensive to manage.

1.6 Limitation of the Research Study

The initial idea was to design a system that would be able to use dynamic form of addressing to store the addresses of the agencies on a Google cloud firebase. But due to financial constrain involved in securing full functionality of the firebase from Google and the complications in making

the various components of the system to communicate, static addressing was later adopted. Dynamic addressing would allow the emergency centers to move around and still receive notification of an accident based on their distances from the accident location. In addition, the system was tested with a radio controlled toy car which cannot give a perfect simulation of accident.

1.7 Definition of Terms

Alert System: A system that can send notification out to targeted persons or devices when it is triggered.

Global Positioning System (GPS): A satellite and radio navigation based system that is owned and regulated exclusively by USA Air force.

Accelerometer: Is an instrument that is used to measure moving or vibrating body acceleration in all directions.

Smart Phone: A mobile phone that performs like a computer in many different ways, typically, having internet access, a moderate to large touch screen and an operating system that can run applications that are downloaded from different sources.

Firebase: Is a cloud-hosted NoSQL real-time database that enables information to be stored and synchronized in real-time.

Mobile Application (App): These are developed computer programs or software application that are designed and are capable of running on devices that are mobile such as phone.

False Negative: An instance where the mobile-phone does not detect a crash even though a crash has occurred

False Positive: An instance where the mobile phone detects a crash even though it has not occurred

True Positive: Number of actual crash instances that the system classified as crashes

True Negative: Number of non-crash instances that the system classified as false crash

CHAPTER TWO

2.0 LITERATURE REVIEW

2.1 Review of Some Related Literature

Crashes especially those involving loss of lives around the world and in particular developing countries is alarming, (Azeez *et al*., 2015). In 2014, W.H.O reported that, even though road crashes are sometimes avoidable, vehicle road accidents caused the undesirable deaths of 1.2 million people worldwide each year and injures about 4 times this number. There are formal and informal ways of reporting road crashes. Most often, good Samaritans report road crashes formally to the concerned agencies, through various channels, which include physical reporting to the road traffic office or telephone call to the road traffic emergency numbers. Studies have shown that delay in reporting a road crash to the appropriate agency and the time it takes the rescue agencies to arrive at the accident location is primarily the biggest reason why causality rate is high in serious road crashes, (Aloul *et al*., 2014).

Casual inquiries from the concerned agencies indicate slow and cumbersome methods of crash detection, reporting and locating the scene of the road crash. For this purpose, some vehicle manufacturing companies embed accident detection systems in their vehicles. These systems use impact sensors to generate impact signals that are fed into Geographical Positioning System (GPS) receivers, which communicate with GPS satellites. In this information age, almost every individual owns a mobile phone; especially smart phones. Smart phones can be found everywhere and with network connectivity, its in-built facilities can be utilized to develop a lot of user applications. Recently released models of smart phones support sensors including audio recorders, GPS and accelerometers in addition to much other functionality, (Aloul *et al.*, 2014).

Most smart phones have wireless information facilities that provide extra possibilities to build consumer apps that take advantage of the sensors and network connectivity that the different kinds

of. Therefore, impact signals generated using in-phone accelerometer and GPS can be used to provide real-time and adequate method of reporting and locating road crash scene. With the location tracking capabilities of GPS device, it becomes easier to generate location coordinates of a crashed vehicle.

Yogesh and Rane (2014) reviewed the various accidents detection systems and proposed an embedded system. The system proposed was Advanced RISC Machines (ARM) based accident detection using GSM, GPS and Micro Electro-Mechanical Sensor (MEMS). Some steps that will be helpful in reducing the rescue time were outlined. These steps are firstly, fast and accurate accident detection and reporting to a Public Safety Answering Point (PSAP). Secondly, fast and efficient evacuation of occupants trapped inside a vehicle. The proposed system used sensors like opt coupler, ultrasonic, alcohol sensors and subsystems like MEM. The system is hardware based and the entire work has to be integrated with the automobile to validate its functionality and reliability which will be expensive to achieve.

Khalil *et al.* (2018) proposed automatic road accident detection using ultrasonic sensor that can detect when a car is close to an object which could be another car. The ultrasonic sensors are placed inside the car on two positions, the front windscreen and the back windscreen. Two thresholds were set for both of the sensors. As the care gets near an object, the distance between the two sensors decreases and this immediately set an alarm. GPS was used to find location of the car is. The system relied majorly on the sensors to avoid crash; however, sensors are not 100 percent reliable because they can fail any time.

Beramy *et al*. (2016) proposed accident detection and alerting system that detects road crashes, gets the location of the crash and sends an SMS containing the location to a hospital or police station. The system is a hardware built on a microcontroller board (Arduino board), with a GPS module and a GSM module attached to it. The system uses a 3-axis accelerometer to detect accidents. When an accident is detected the micro-controller gets the coordinates from the GPS and uses the GSM modem to send the notification message. The modules are first turned on, then the GSM module acquires network signal then the system goes into a standby mode. When an accident occurs, the impact sensor gives a positive output and the micro-controller acquires the coordinate's location using the GPS module. The coordinates are integrated into the notification message and then sent to emergency services and contacts predefined by the user. The system is hardware based; retrofitting the designed system for older vehicle will be very expensive,

Narendar and Tejar (2013) developed a system that can detect a vehicle crash at the same time keep the driver of the vehicle on a serious alert if the speed limit is exceeded, using Micro Electromagnetic System (MEMS) to detect road crashes, Radio-frequency Identification (RFID) to detect various zones, a GPS module to get location coordinates and a GSM module to send accident details and location and also the vehicle and speed limit details to traffic police was developed. The main idea was to come up with a controller that has a smart display that is meant for vehicles speed limit and crash alerts which can run on an embedded system. When an accident occurs, MEMS sensors begins the process, then the micro-controller triggers and activates the module of the GPS to get the location of the vehicle and also the GSM module is activated to send the SMS. The sensors and other components that are needed for the design are expensive to acquire.

(Njuguna, 2012) developed a vehicle accident detection and reporting solution that is an Instant GPS motor based. The system consisted of components such as accident detection component, vehicle tracking component and accident surveillance module; the accident detection module has impact sensors, signal conditioning electronics and accident detection logic unit. Active GPS device with three important capabilities including a GPS signal interface, digital input interface and a wireless interface with transmitting capabilities make up the vehicle tracking module. The surveillance module has a computer with an input interface for a wireless GSM enabled mobile phone. The computer hosts a database for recording accident incidents, accessible through a website, the recorded accident data is displayed at the client workstation against a GIS digital map backdrop, using customized software to authorized users of the system via the website and the internet technology. New accident incident will sound an alarm at the client side to alert the user. This enables authorized users, for example the police and ambulance paramedics to learn about an accident occurrence, allowing them to track and move to the accident site for evacuations. The system relied on too many components like the website, digital interface, and signal conditioning electronics to function.

Dang *et al*. (2016) designed automatic fall detection system using smart phone acceleration sensor that works with the accelerometer of the smart phone. A threshold was set for the accelerometer. The system was meant to detect if there is an accidental fall that will be strong enough to trigger the acceleration threshold that was set. The system was tested and was meant for old persons or people with ailment that needed to be monitored constantly to avoid dangerous fall. The smart phone will be attached to a person's body, in an eventual fall, an alarm will be triggered and the

Smartphone will make sound for as long as the alarm clock is set. The system is only designed to detect a fall; it is not designed to notify emergency responders in a situation where the fall is fatal.

Kaladevi *et al.* (2014) developed and implemented an accident detection system using an android smart phone. This system uses the heart beat sensor (IR sensor and microphone) attached to the seat belt to measure the heart beat rate, the measured data is then transmitted to the phone using Bluetooth. This is achieved via the control unit (micro-controller) which is connected to the heart beat sensor and is also installed in the vehicle. When there is variation from the normal heart beat rate, then an accident is detected. The android smart phone then uses its inbuilt GPS to get the crash location and then send the message to an emergency contact. However, heart beat rate is not entirely reliable because it cannot be controlled in certain circumstances.

Accelerometers and acoustic data when properly used by smart phones can detect traffic accident and send notification and situational awareness via coordinates from GPS to a central server created for emergency personnel. The system developed is WreckWatch; it was built for android smart phones. It has two phases a client side installed on a smart phone which detects accident, show map and gives provision for registering emergency contacts, and also gives room for eyewitness report to provide situation awareness data, and a server side this offers aggregation of information and a means of communication for emergency staff, family and friends. It also enables customers to submit accident features (such as acceleration, path, and velocity) and provides multiple interfaces, such as any Google map and Extended Mark-Up Language/JavaScript Object Notation (XML/JSON) services for web and for accessing the provided information, (White *et al.*, 2011).

2.1.1 Android Smart Phones

These are the smart phones that run on the android operating system. They can perform the typical computer functions, access the internet and have a moderate to large touch screen. The operating system is capable of running downloaded Apps. Smart phones has number Sensors that your software can use, such as accelerometer, magnetometers, proximity sensors, barometer and gyroscope and support wireless communications protocols such as Bluetooth, Wi-Fi, and satellite navigation, (http://www.wikipeadia.com).

2.1.1.1 Some Key Features of Android Smart Phones

i. Central Processing Unit (CPU)

The CPU of smart phones is optimized to perform typical functions of computer. The output of the portable CPU relies on what you want, not just on the clock rate but also on the memory hierarchy. They can effectively perform in a low power environment, (http://www.wikipeadia.com).

ii. Display

The screen is one of the main features of android smart phone. Depending on the device design, the screen fills most or nearly all of the space on a device front service, (http://www.wikipeadia.com).

iii. Battery

The battery life of smart phone has gradually been improved upon depending on the model of the phone. Earlier smart phones battery could not handle significantly the power requirements of the embedded processor and color screens.

iv. Location detection Ability

Smart Phone has the ability to easily detect your location, and even keep track of your movements. This location detection is achieved with the inbuilt GPS technology on the smart phone, (http://www.wikipeadia.com).

2.1.2 How GPS is working

GPS with the help of satellites transmits very accurate signals, enabling GPS receivers to calculate and show the user's exact location, velocity and time data. GPS receivers can use the mathematical principle of trilateration to determine your location by capturing the signals from satellites. With the addition of computing power and data stored in memory such as road maps, points of interest, topographic information, and much more, GPS receivers can convert data about place, velocity and time to a helpful display form, (Zahradnik, 2017).

GPS is a network of approximately 30 Earth orbiting satellites at 20,000 km altitude. The system was initially created by the U.S. government for military navigation, but now anyone with a GPS device, whether it is a SatNav, a mobile phone or a handheld GPS unit, can receive the satellite radio signals.

Wherever you are on the planet, there are' noticeable' at any moment at least four GPS satellites. Each transmits data at periodic intervals about their place and the present moment. Your GPS receiver intercepts these signals, traveling at the velocity of light, which calculates how far each

satellite is based on how long it took for the messages to arrive. Once you have data about how to do it, Institute of Physics (I.O.P, 2014).

2.1.2.1 GPS Trilateration

Imagine you are standing somewhere on Earth with three satellites in the sky above you. Figure 2.1 shows GPS trilateation. If you understand how far you're from satellite A, you know you have to be on the red circle somewhere. You can work out your place by seeing where the three circles intersect if you do the same for satellites B and C. While using overlapping spheres rather than circles, this is just what your GPS receiver does. The more satellites above the horizon, the more correctly they are, (I.O.P, 2014).

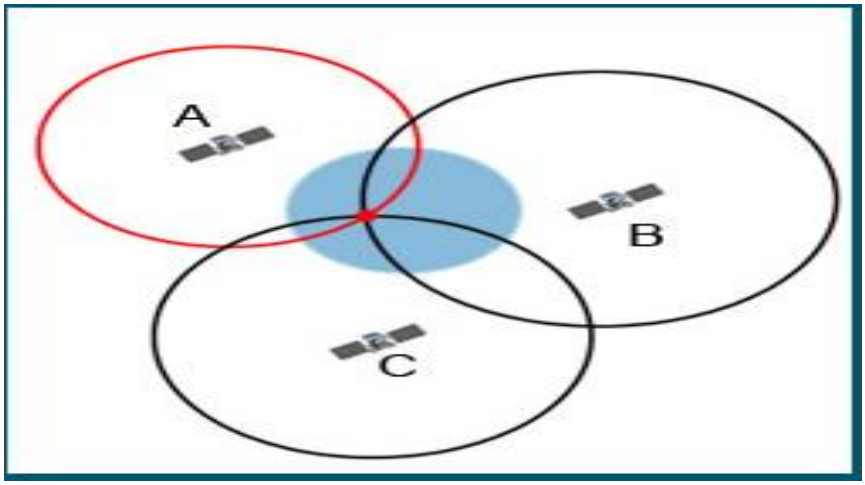


Fig.2.1: GPS Trilateration Illustration (Source: I.O.P, 2014).

2.1.2.2 Limitations of GPS

Barriers like thick forest, canyon walls, skyscrapers, bridges can block the GPS signal by making it hard or impossible to navigate accurately. Similarly, indoor and underground areas, GPS does not operate well. Maintenance of satellites, radio interference and solar storms can cause gaps in coverage.

2.1.3 Difference between GPS and GPRS

GPS stands for Global Positioning System whereas GPRS stands for General Packet Radio Service. In simple terms, GPS will give you the Latitude and Longitude coordinates location while GPRS will allow you to transfer data over cellular networks.

GPS technology can identify any Earth position or address. Although it involves the operation of various satellites, it can be used almost anywhere in the globe, making it highly available and advantageous for companies with many distinct geographical places. GPRS is the wireless information service most frequently used. Older versions relied on 2G cellular networks, but now most use 3G technologies. GPRS enables cellular devices to perform functions such as multimedia messages and Internet surfing, (GB Blog Official, 2018).

2.2 Evolution of 2G, 3G and 4G Networks in Mobile Phones

Mobile devices need a strong and reliable cellular network in other to be able to effectively use the GPRS facility built on them. For the system to work, a good network type is needed based on some certain network parameters such as speed, bandwidths, frequency, and accessibility. All android smart phones have 2G and 3G. Basically, to use any of the network type, the owner of a smart phone must choose between the network types. 4G network have higher speed and bandwidth but are not supported by all the cellular network service providers,

2.2.1 Second Generation Mobile Telecommunication (2G)

2G networks started to appear around the late 1980s and in many nations these networks arrived in the early 1990s. In contrast to its predecessor which was the 1G network, Digital modulation methods were used in these technologies, resulting in superior speech quality. But the circuit stayed turned to the networks. 2G got fresh facilities like SMS, fax, and Web Appilication Protocol (WAP). Encryption was introduced, that greatly improved safety and solved most 1G security issues, as well as improved service quality for error detection and correction. Some 2G systems include the Global Mobile Communications System (GSM), IS-54 (digital AMPS), IS-95 (CDMA), GPRS and EDGE, (Tanko, 2014).

While 2 G technologies significantly enhanced mobile communications, leading to an explosion in numbers of subscribers, it was with many constraints. One of these includes the fact that 2G was a circuit-based network, so it utilizes bandwidth and resources inefficiently, which greatly limits the capacity of elevated information rates, is also unable to manage complicated information such as video and also limits the amount of customers. Other limitations include the lack of interoperability between 2G networks, poor standardization and the fact that 2 G provides very limited services and apps, (Tanko, 2014).

2.2.2 Third Generation Mobile Telecommunication (3G)

3G Mobile networks have been created based on the International Telecommunications Union (ITU), a unified family of norms capable of working together and meeting International Mobile Telecommunications (IMT-2000) specifications, to construct mobile networks that deliver multimedia services and other services accessible on wireless devices. These networks began operating mainly around the beginning of the 2000s. 3G technologies used circuit switching for

voice/SMS and packet switching for data services. The technologies include Wideband-Code Division Multiple Access (W-CDMA), Code Division Multiple Access (CDMA-2000) and Time Division-Synchronous Code Division Multiple Access (TD-SCDMA). This network operated on the 2100MHz frequency band and provided greater speeds at high mobility from 144kb / s to 384kb / s and low mobility from 2Mbps. 3G increased network capacity to meet up with demand and actualized global roaming for subscribers.

3G has been a tremendous success, particularly in terms of standardization, but there are constraints and expectations that exceed it. Such issues include the high spectrum license price, high 3G network cost that makes most operators return to 2G. There are also problems with delayed roll-out and patchy coverage. In addition, with the latest fast evolution of information systems and services, mobile devices requiring high mobility, much greater data rates and interoperability, it is necessary to harmonize all network technologies with much greater data rates at any time, (Tanko, 2014).

2.2.3 Fourth Generation Mobile Telecommunication (4G)

Soon after 3G, mobile telecommunications networks of the fourth generation are technologies constructed to meet the set of norms set by the IMT-Advanced requirements. These networks will reach elevated mobility speeds of 100Mbps and low mobility speeds of up to 1Gbps. This allows wireless systems to reach the current capabilities of wireless systems and to trigger a mobile broadband. Also the 4G network is to be an ‘open wireless’ system which means it should be a network with a unified core which is accessible from different wireless(access) technologies, this is aimed at harmonizing and further standardizing all the available wireless technologies. 4G networks are also ‘all IP’ and fully packet switched networks, (Tanko, 2014).

2.3 Google Firebase Cloud Systems

Firebase Cloud Messaging (FCM) is a cross-platform messaging solution that enables you to deliver emails reliably and free of charge. Using FCM, you can notify a client app that you can synchronize fresh email or other information. To drive user re-engagement and retention, you can send notification emails. A message can transfer a payload of up to 4 KB to a client app for use in instances such as immediate messaging, (https://www.firebase.google.com/docs/cloud-messaging).

2.3.1 How Firebase Works

There are two primary elements for sending and receiving an FCM application:

i. A trusted environment like Firebase Cloud Functions or an App Server for building, targeting and sending emails.
ii. An iOS, Android, or web (JavaScript) client App that receives messages.

Use the Firebase Admin Software Development Kit (SDK) or the FCM server protocols to send emails. You can also use the Notifications composer to test or send marketing or commitment posts with strong integrated targeting and analytics..

Use the Firebase Messaging API and Android Studio 1.4 or greater with Griddle to write the Firebase Cloud Messaging Android client app. The guidelines on this page suppose that the measures to add Firebase to your Android project have been finished. FCM customers involve Android 4.1 or greater phones that also have the Google Play Store App installed, or an Android 4 emulator.

2.4 Java Application Programming Interface (API)

Java Application Programming Interface (API) is a list of all courses in the Java Development Kit (JDK). It involves all java packages, classes, and interfaces as well as their techniques, areas, and builders. These pre-written courses give a programmer a great deal of functionality, (http://www.wikipeadia.com).

2.5 Android Studio

Android Studio is Google's official integrated development environment for the Android operating system, built on IntelliJ IDEA software from JetBrain and specifically intended for Android development. It can be downloaded from Windows, macros and the operating system based on Linux, (http://www.wikipeadia.com).

Android studio allows the user to apply changes and push code and resource changes the running App and in some cases, without restarting the current activity. It also has features like the intelligent code editor for writing of better codes, fast emulator to starts Apps. Android studio also comes with templates and sample Apps which includes projects and code templates that allows the addition of well established patterns such as navigator drawer and view pager.

One of the most exciting features of the Android studio is the Firebase and cloud integrations. The firebase assistant allows a developer to connect Apps to firebase and add services such as analytical, authentication and notifications with step-by-step procedures.

CHAPTER THREE

3.0 RESEARCH METHODOLOGY

3.1 Methodology

The proposed system will perform both accident crash location detection and notification. The system design block is illustrated in figure 3.1.

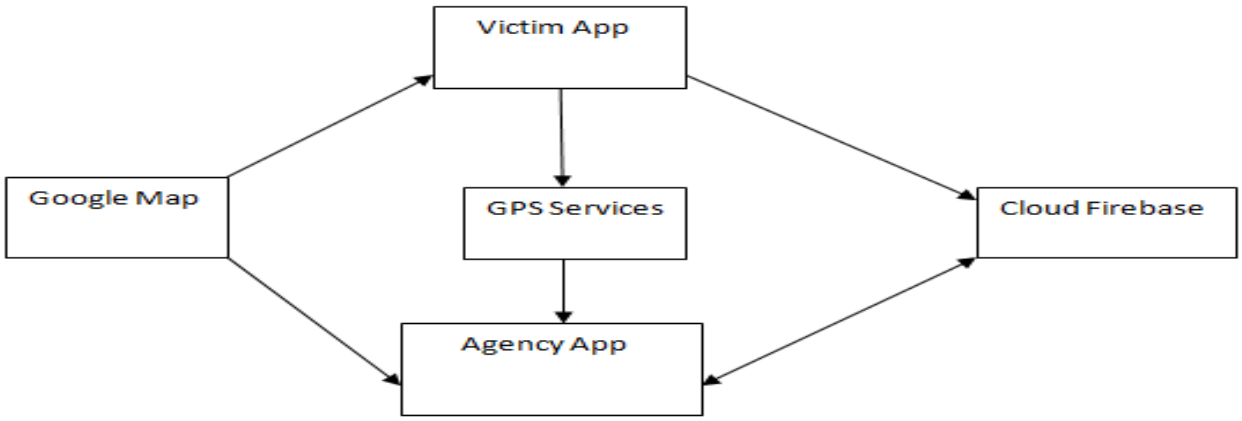


Fig. 3.1: Block design diagram.

The system is made up of two App's and a Google firebase cloud system for registering, storing and retrieval of information of the agencies and victim. The two Apps are the crash detector called user or victim's App and the agency location called agency App. Both Apps were developed using Android studio Integrated Development Environment (IDE). The victim's App uses the GPS technology and services to access the Goggle map. A Google cloud Firebase which store information on database in real-time was used to store the phone numbers and the geo-location coordinates of the agencies. The victim's smart phone with the App detects the crash, gets the geo-location coordinates through the GPS services, retrieves the agency phone number from the Firebase and notifies the closest agency to the accident site. The agency App is only used to register an agency on the Google Firebase cloud system.

3.2 System Architecture

The system architecture shown in Figure 3.2 has two parts, the client side - the mobile-phone (locating and reporting device) and the cloud system which act as a server. The smart phone on which the victim App was installed is inside the moving vehicle. As the vehicle accelerates and crashes, the victim android smart phone in conjunction with the App detects the crash if the acceleration threshold set for the App is triggered and gets the location coordinates from GPS satellite and the link to the Google map image of the crash site from the Google map server in real-time.

The victim App then access the real-time firebase on the cloud system, fetch both the location coordinates and mobile phone number of the any of the three registered agencies that is closest to the accident location, then sends SMS containing a short message with the link of the Google map image to the nearest concerned agency phone number. The staff of the concerned agencies has full access to the features of the system such as adding or deleting an agency and accessing details of road crashes on the firebase.

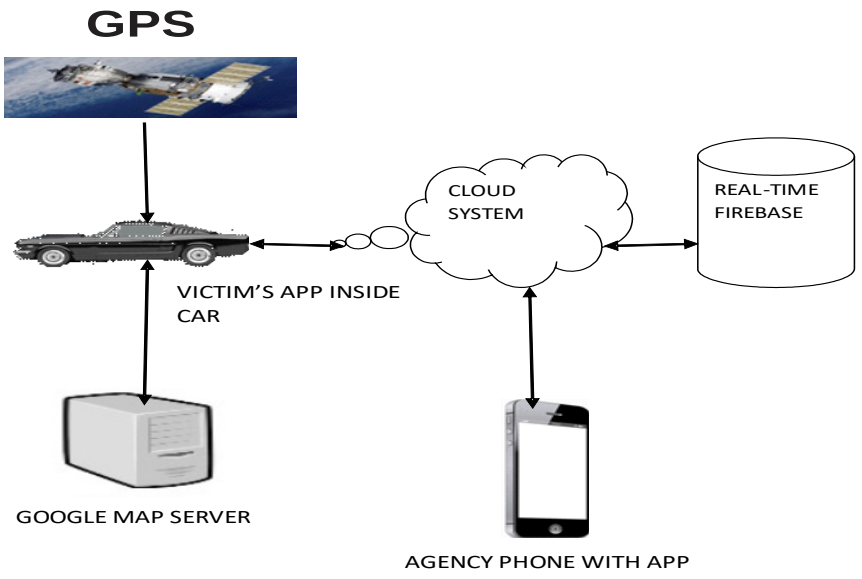


Fig. 3.2: System Architecture

3.3 System Design

The two Apps, victim and agency App were developed and implemented using Integrated Development Environment (IDE) of Android studio, with Java programming language for the Events and XML for the interface design. The implementation was strictly on Android smart phones with minimum Application Programming Interface (API) level 16.

The system design is illustrated with the Entity-Relationship diagram in Figure 3.3; it is made up of five Entities- the Victim's App, the Agency App, the Firebase, the Concerned Agency and the User. The entities are denoted by rectangle. The entities; User, Concerned Agency and Firebase have attributes that are denoted by a circle. The User has email, phone number, name, date of birth, longitude and latitude as attributes. The Concerned Agency has agency name, email, password, phone number, longitude and latitude as attributes and the Firebase has gmail and password as attributes.

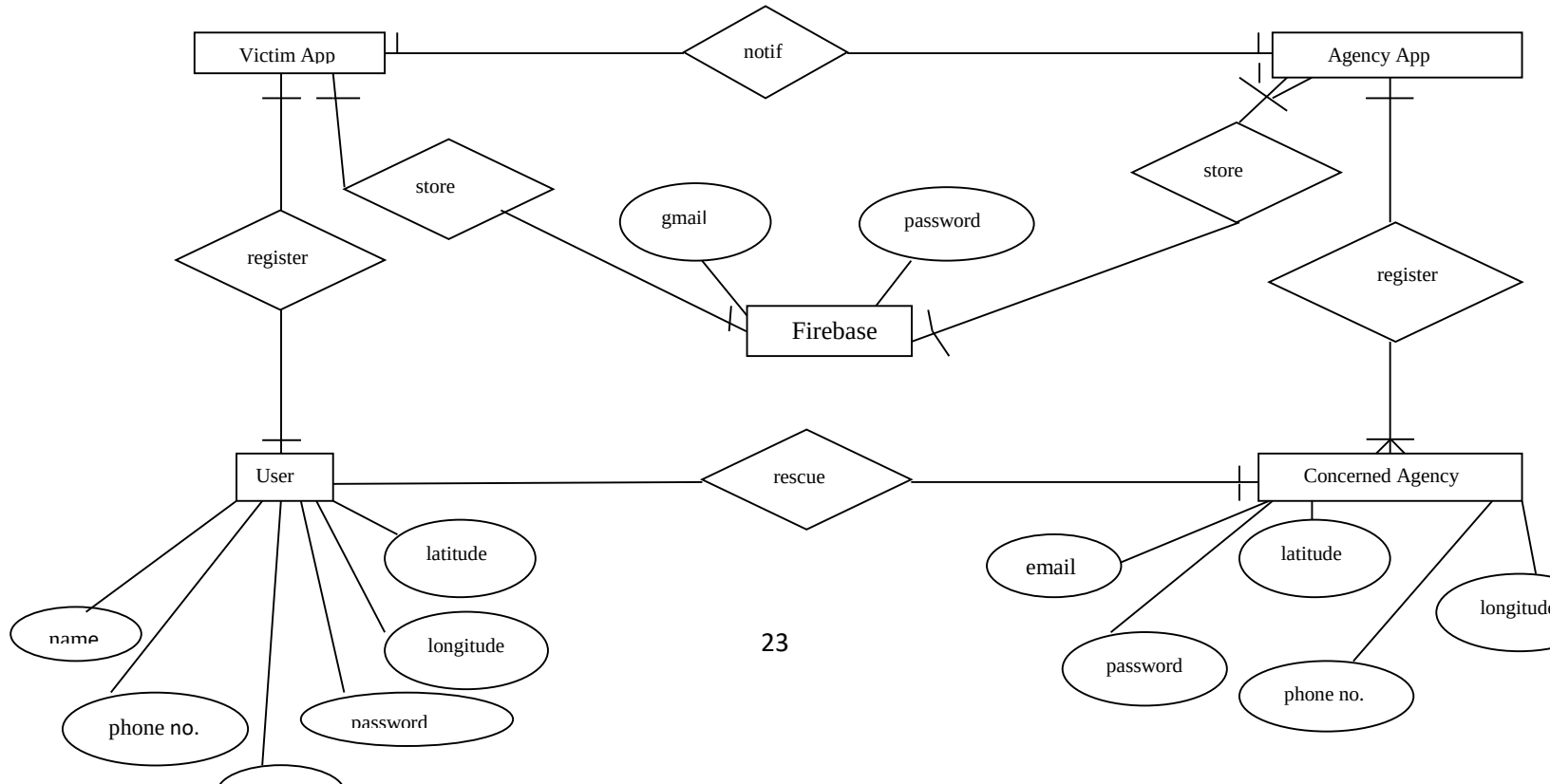


Fig. 3.3: System ER-

The relationships between the Entities are denoted by the diamond shapes. The User has register as a one-to-one Relationship with the Victim App, a relationship called store with the Firebase which is also one-to-one. The Victim App has a relationship notify with Agency App which is one-to-one too. The Agency App has a relationship with the Firebase called store which is one-to-many. The Concerned Agency has two Relationships. One with the User called rescue which is one-to-one and one with the Agency App called register which is a many-to-one Relationship.

3.3.1 The Victim App (Crash Detector)

The victim App was designed using android studio integrated developments environment. The interfaces of the App were designed with XML and java programming was used for the events. Pre-defined classes and methods were used to enable the App access GPS facility. For the victim smart phone to be able to detect a crash, an acceleration threshold was set for it through the App. This is the acceleration at which the mobile-phone detects a crash. If the acceleration experienced by the mobile-phone exceeds this threshold, then it concludes that there has been a crash. Intuitively, if the threshold is set very high, it can lead to an instance where the mobile-phone does not detect a crash even though a crash has occurred and a very low threshold might lead to false positive. Therefore, to set a threshold for the Victim's App, proper analysis is paramount.

The acceleration threshold for airbag deployment is 60Gs (around 600m/s^2), (White *et al.*, 2011).

This threshold eliminates any chance of a false positive, but it ignores low speed crashes. Setting a 60G threshold will be too high for it to detect crashes in which the impact is not high enough to deploy airbags. This is because the threshold is a function of the overall vehicle design. Air bags are triggered by the accelerometer attached to the vehicle. These accelerometers are physically installed on the car's chassis, which means that their movement directly mirrors that of the vehicle and thus experiences every force encountered by the vehicle. Smart phones, however, are likely to be held in a pocket or in a cup holder. Car safety systems are designed to reduce the force on the occupants of the car during an accident and because of this, the forces and acceleration experienced by the mobile-phone is significantly less than the forces experienced by the accelerometers in the cars, (White *et al.*, 2011).

Due to the physical environment of the mobile-phone, the threshold to be set should be significantly lower than the speed needed for airbag deployment. White *et al.* (2011) used a threshold of 4Gs, which is approximately 40m/s.^{2}To set the acceleration threshold of the victim's App, the android smart phone accelerometer designed for the App will have to be integrated with the android device to detect any form of motion. The excerpt from the code that allows this integration and makes the accelerometer of the smart phone enable in other to detect a crash is outlined in figure 3.4 from the code excerpt, the acceleration threshold set for the victim App was 15G.

CrashDetection

```
public class SensorService extends Service implements SensorEventListener
{

   // TAG to identify notification

   private static final int NOTIFICATION = 007;

   private Sensor accelerometer;

   private SensorManager mSensorManager;

   private double accelerationX, accelerationY, accelerationZ;

   private ir
```

Fig.3.4: Code excerpt for crash

```
      accelerationX = (Math.round(sensorEvent.values[0]*1000)/1000.0);

      accelerationY = (Math.round(sensorEvent.values[1]*1000)/1000.0);

      accelerationZ =

      if (accelerationX > threshold || accelerationY > threshold || accelerationZ
> threshold) {
```

3.3.2 The Agency App (Agency Locator)

The agency App was also designed with android studio like the victim's App. The App was only used to register each of the three agencies that were used to implement the system. The information supplied by the user and the agencies using the victim App and the agency App respectively were all stored on the Google cloud firebase.

3.3.3 Google Firebase Cloud System for the Victim and Agency Apps

The Firebase Cloud Messenger and the Software Development Kit were created through these steps.

i. Firebase was added to the Android project.

ii. Dependency for the Cloud Messaging Android library was added to module (app-level) Gradle file (usually app/build.gradle): implementation 'com.google.firebase:firebase-messaging:17.6.0'

iii. The victim's App manifest was edited, This is necessary if you wish to handle any messages beyond receiving background notifications on Apps such as receiving notifications in fore grounded Apps, receive data payload, send upstream messages. To access the device registration token when the application is initially started, the FCM SDK generates a registration token for the App instance. The token can be retrieved. Because the token could be rotated after initial startup, latest updated registration of the token is token taken. The registration change when:

i. The App deletes Instance ID

ii. The App is restored on a new device

iii. The user uninstalls/reinstall the App

iv. The user clears App data.

The Google firebase account was accessed through a Google mail account. The Firebase uses JSON (JavaScript Object Notation) to store data. JSON is an open-standard file format that uses human-readable text to transmit data objects consisting of attribute–value pairs and array data

types. The data are stored in a Tree-like structure. Figure 3.5 and 3.6 shows the firebase with three agencies information and victim information respectively.

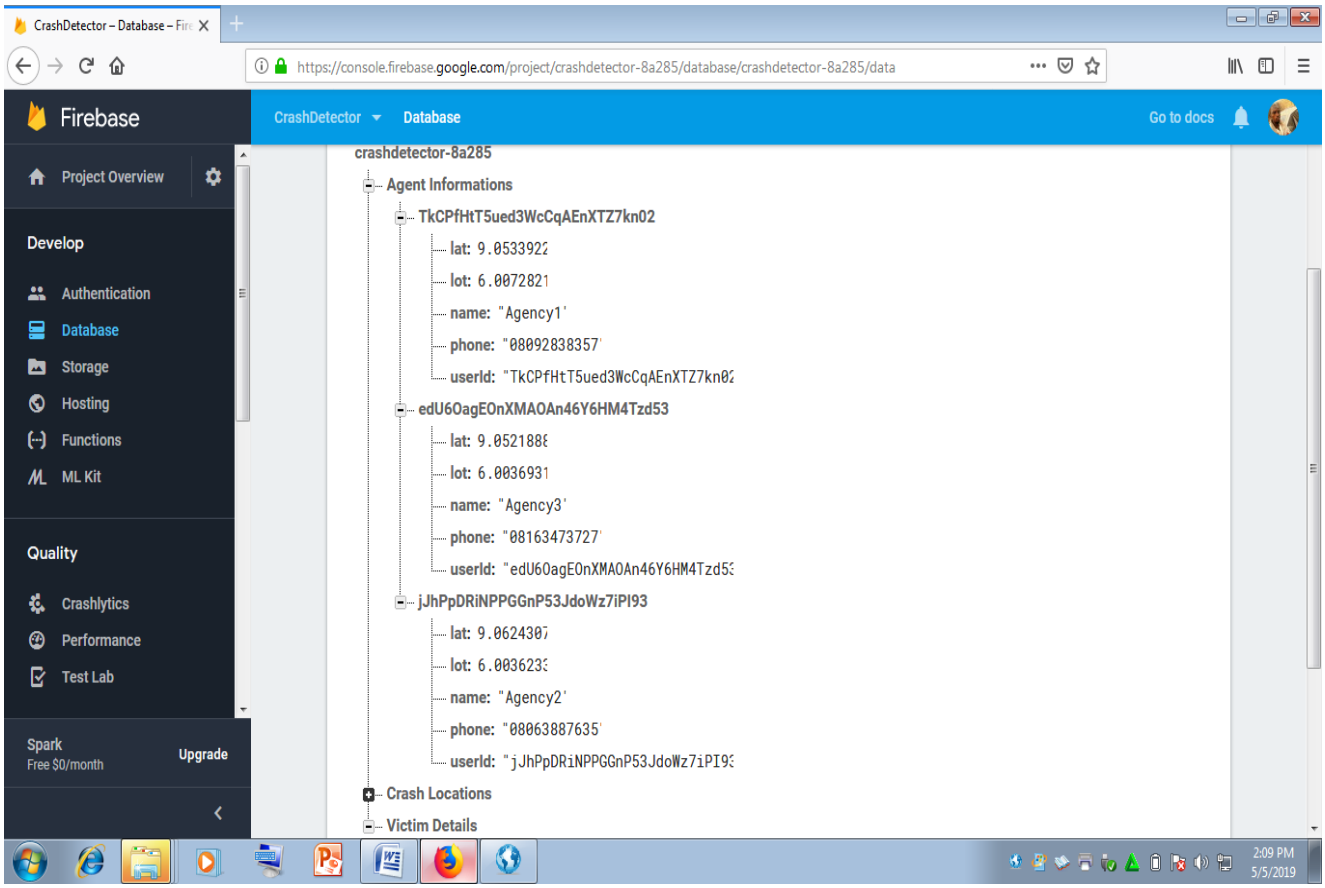


Fig. 3.5: Firebase with three agencies information

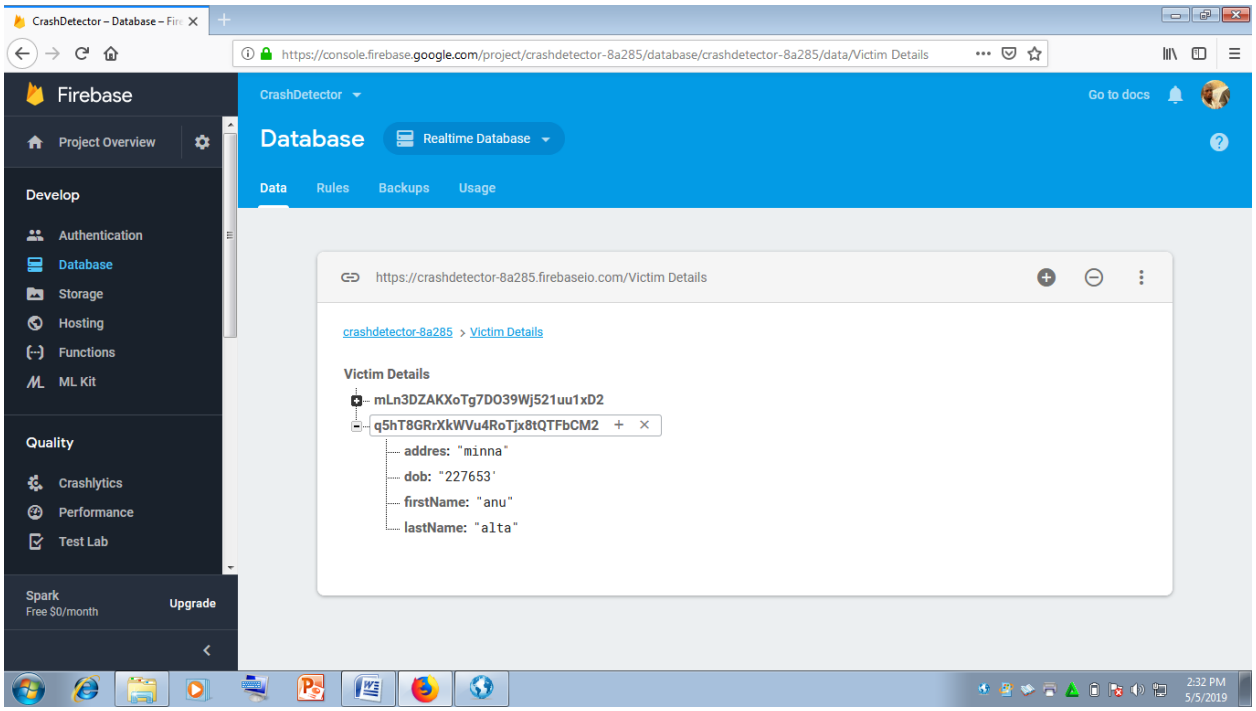


Fig. 3.6: Firebase with a victim registration details

3.3.4 Obtaining Google Map from the Google Server

The majority of android smart phones come with Google Map Apps pre-installed. The Google Maps App is activated by turning on the “Location” finding feature on the pull down menu. The Google Map relies on the smart phone’s GPS settings to know where its location is. The smart phone location actually comes with the graphical image of the location as supplied by the Google Map server.

3.3.5 Crash Location Reporting

The crash is both detected and reported by the victim’s App. When the acceleration threshold of the victim smart phone is triggered; the victim’s App retrieves the stored information of the closest concerned agency from the Google firebase, and send an SMS to it. Three concerned agencies data were stored on the Firebase. From the code excerpt shown in figure 3.7, the three agencies were

created as variables and named as agency_A, agency_B and agency_C. The mapping activity is done by the algorithm by comparing the location coordinates of the three agencies with the captured location coordinates of the crash site and send SMS to the concerned agency phone number that is closest to the crash site. If the agency registered is only one, then only the registered agency will receive the alert. The mapping comparison starts when the agencies are two or three but not more than three.

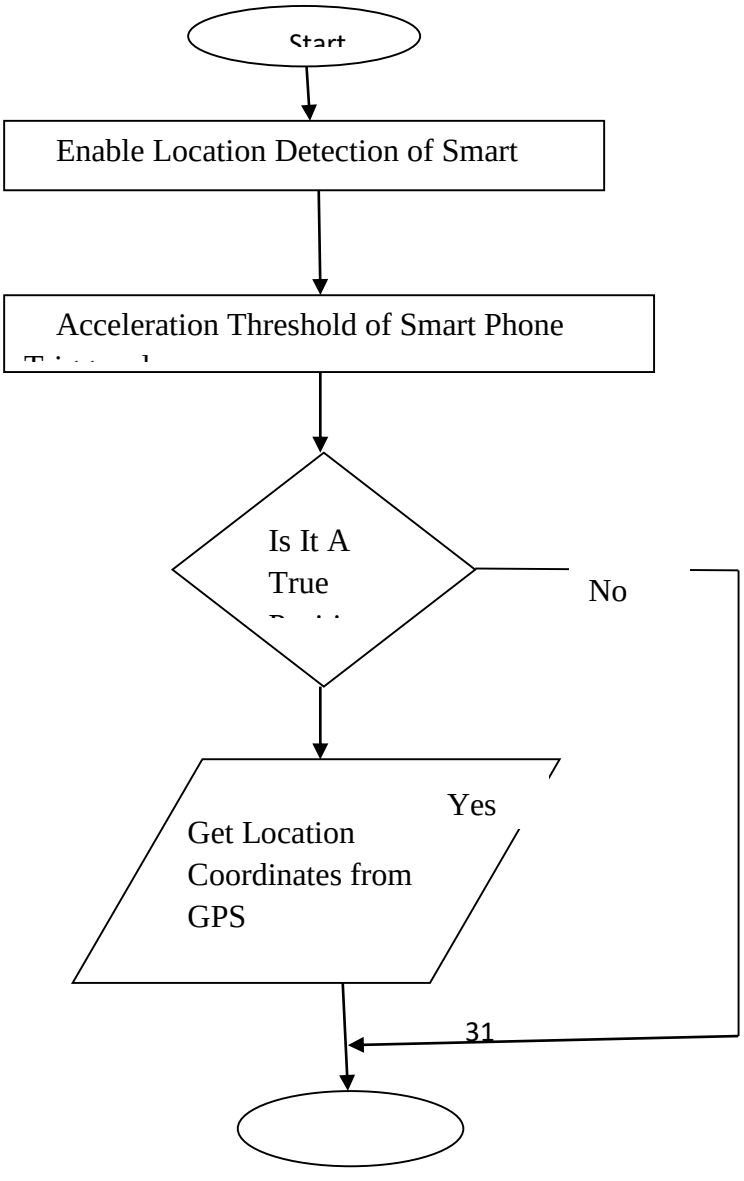
```
// CrashReporting
if(agency_A< agency_B && agent_A< agency_C){
    sendSMS(victim,ph1,crash,agency_A);
    saveToFirebase(crash);
}else if(agency_B<agency_A && agency_B<agency_C){
     sendSMS(victim,ph2,crash,agency_B);
      saveToFirebase(crash);
                    .          .
}else if(agency_C<agent_B && agency_C<agency_A){
    sendSMS(victim,ph3,crash,agency_C);
   saveToFirebase(crash);

}else if(agency_A == agency_B || agency_A==agency_C){
    sendSMS(victim,ph1,crash,agent_A);
   saveToFirebase(crash);
}else if(agency_B == agency_A || agency_B==agency_C){
```

```
            sendSMS(victim,ph2,crash,agent_B);
            saveToFirebase(crash);
        }else {
```

Fig.3.7: Code excerpt for crash reporting

3.4 System Flowcharts

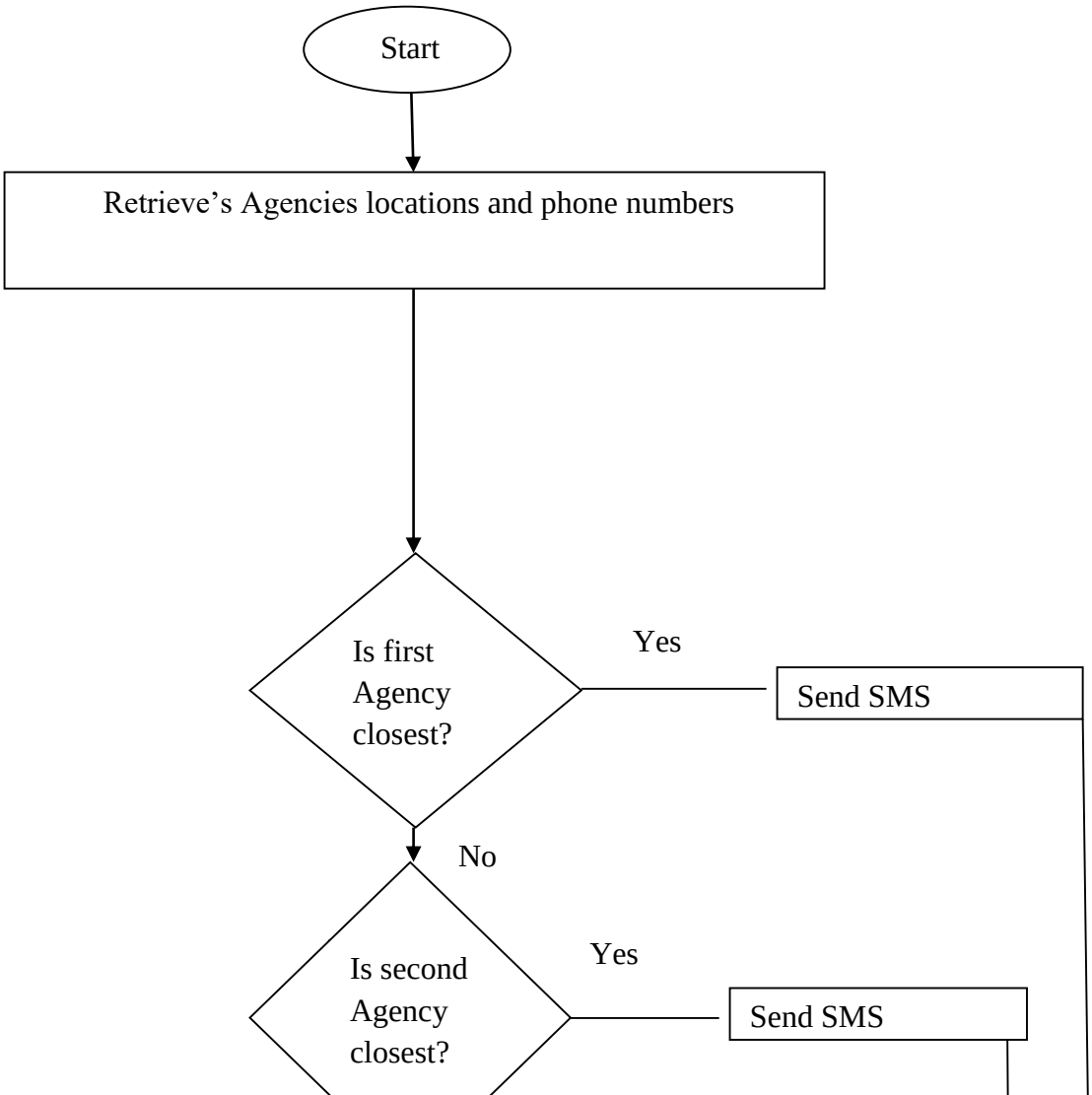
The location detection and location reporting activity of the system are represented with two system flowcharts. Figure 3.6 is the System Flowchart for the crash location detection.



Ye

Fig. 3.8: Crash location detection

After detecting the crash, the victim's App retrieves information from the firebase and sends the notification in form of SMS to the nearest agency. Figure 3.9 shows the mapping activity of the algorithm.



CHAPTER FOUR

4.0 RESULTS AND DISCUSSION

4.1 Results Obtained From the System

The results obtained from the system were analyzed. The system was tested first before the result analysis was carried out.

4.2 Testing the Proposed System

The proposed system was tested with the unit test and the system test approaches.

4.2.1Unit Test

Unit test was carried out after each of the system modules where completed. The modules are the victim App, the agency App and the Google firebase. The unit test was important to check for likely errors and the communication problem that may surface between the various system

modules. After confirming that the individual units are functioning optimally, the next test approach which is the entire system test was carried out.

4.2.2 System Test

The entire system units where tested together which is the system test. The system test required the set up for the experiment that was carried out.

4.2.2.1 Experimental Setup for the System Test

The following items were used to test the system.

i. Four Android Smart Phones, one had the Victim App installed on it, and each of the remaining three had the Agency App installed on them.
ii. A laptop system used to access the Google Firebase account created
iii. A modem for internet connectivity for the laptop
iv. A Radio-controlled toy car.
v. Masking tape

Table 1 show the model and properties of the four android smart phones used for the experiment

Table 4.1: List of the Android Phones used and their properties

Phone Name	**Model**	**Software Version**	**Internal Storage**	**Network Type**			**App installed**	
				2G	**3G**	**4G**	**Victim's App(3.14MB)**	**Agency App(2.72 MB)**
Nokia 5		7.1	16G	✓	✓	✓	✓	

INFINIX X559C	7.0	32G	✓	✓	✓	✓	✓
Techno LA6	7.0	16G	✓	✓			✓
Techno K7	7.0	16G	✓	✓			

A long corridor was used as the traveling path for the radio-controlled vehicle. The path was divided into three sections; each position was 15 meters or more away from the next section. The three smart phones INFINIX 559C, Techno LA6 and Techo K7 were stationed one each on the three positions.

From table 4.1, the three smart phones stationed on the travelling path were the ones with the agency App installed on them. Each of the smart phones on the path represented the location of an emergency center. The agencies registrations were carried out from the various locations the smart phones were stationed in other to be able to obtain there static geo-location coordinates addresses. The victim's registration was done at a random location along the travelling path using the Nokia 5 smart phone.

4.2.2.2 Registration

The victim and agency Apps were used to register a victim and the agencies information's respectively that were stored on the Google firebase. The registration was simple. Figure 4.1 and 4.2 shows the display windows of the both victim and the agency App respectively after successfully installing them.

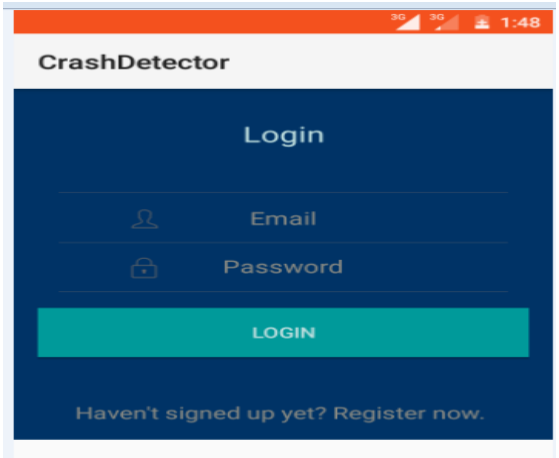
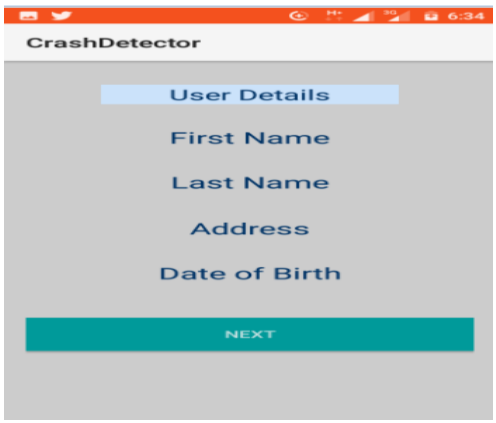


Fig. 4.1: Victim's App display window

Fig. 4.2: Agency App display window

A new user was registered by clicking Register now. After clicking register now, the registration window came up where all the user details that was needed was supplied by the user. For the agency App, the register button was clicked and the agencies information were registered. Figure 4.3 and 4.4 shows the registration windows for the victim and agency Apps respectively.



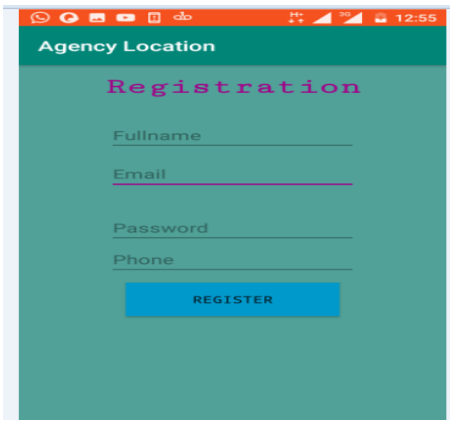


Fig. 4.3: Victim registration window

Fig. 4.4: Agency registration window

Tables 4.2 and 4.3 shows the input specification required for both the victim and the agency App registration respectively.

Table 4.2 Input specifications for the registration window of victim App

User Detail	Data Type
First Name	String
Last Name	String
Address	Alphanumeric
Date Of Birth	Numeric And Special Characters

Table 4.3: Agency App input specification for registration

Agency (Agent) Info	Data Type
Name Of Agency	String
Email	Alphanumeric And Special Characters
Password	Numeric
Phone Number	Numeric

After the registration, the user logged in with a Gmail account, phone number and a password of eight numeric digits. Gmail and numeric password were used to log onto the agency App.

4.2.2.3 Vehicle Crash Setup for the Experiment

The Radio-controlled toy car that was used to simulate the crash has a speed of around 30km/h, Figure 4.3 shows the different views of the vehicle and how the smart phone with the victim App was strapped on the vehicle by a masking tape. The smart phone cellular network type was selected and the data service was switched on; the find location feature was also activated.

Before the vehicle was accelerated for the crash, the victim App was opened on the smart phone and a button was activated to start tracking the location of the smart phone as it accelerates by the

App. The vehicle was accelerated for about 4seconds to attain top speed before it was crashed on a padded surface. The crash was strong enough to trigger the acceleration threshold set.

Fig. 4.5: Radio-controlled car set up for experiment

As soon as the acceleration threshold was triggered, a window shows up with an alarm and a 30seconds timer. The purpose of having the alarm and the timer is to be able to detect fake crash or false positive, a situation where the system detect a crash where there was no crash. The timer counts down from 30seconds, if it is not stopped, after the countdown; the victim's App accessed the firebase and fetches the geo-locations coordinates of the three registered agencies and compare their locations with the location of the crashed vehicle. It then mapped the crash location address to the address of the closest agency to the crash and sends SMS to that agency phone number in real-time. Figure 4.6 and 4.7 shows the window of the countdown timer and the short SMS sent to the closest agency respectively. The SMS contained the link to the Google map server where image of the location of crash site was found.

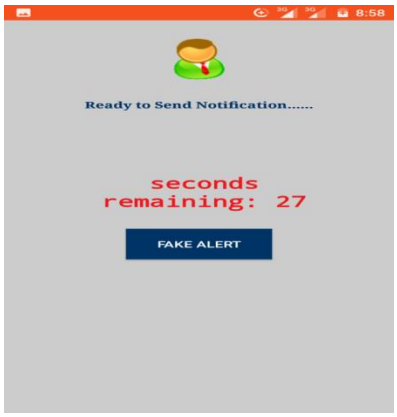


Fig. 4.6: Countdown timer of the

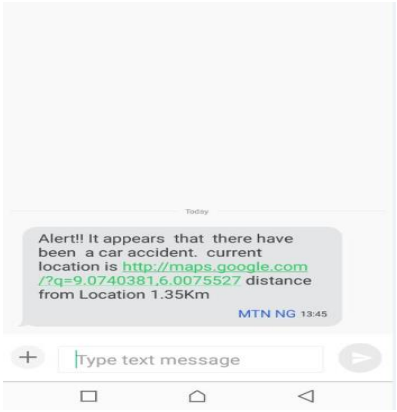


Fig. 4.7: Crash details

The link to the location was clicked and the imagery of the crash location was displayed as shown in Figure 4.6.

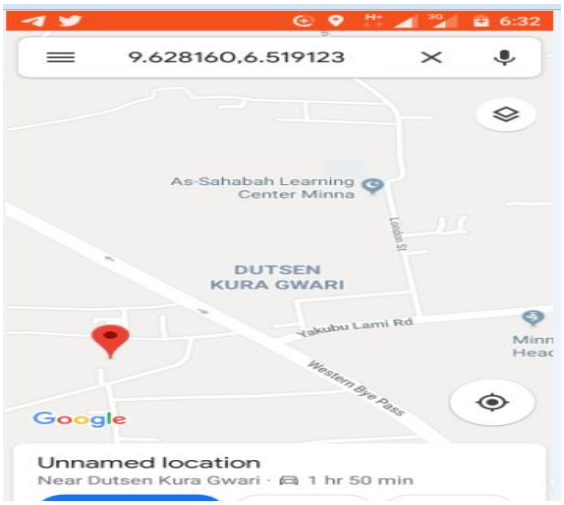


Fig. 4.8: Google map-view of crash location

4.3 Discussion of the Obtained Results

To discuss the result, the system performance was evaluated using two performance indicators. The first indicator used was the confusion matrix. The confusion matrix was used to store the result

of the classifier from 100 crash scenarios. The system detected the class of each instance; this classification is shown in Table 4.4

Table 4.4: The system confused matrix

TOTAL CRASH	**PREDICTED ROAD CRASH**		**TOTAL**
ACTUAL ROAD CRASH	TRUE POSITIVE= 55	FALSE NEGATIVE = 4	59
	FALSE POSITIVE =5	TRUE NEGATIVE = 36	41
TOTAL	60	40	100

The following performance metrics were calculated from table 4.4

i. $$\text{Accuracy} = \frac{TN + TP}{TP + TN + FP + FN} = 91\%$$

ii. $$\text{Sensitivity} = \frac{TP}{TP + FN} = 93\%$$

iii. $$\text{Precision} = \frac{TP}{TP + FP} = 92\%$$

iv. $$\text{Specificity} = \frac{TN}{TN + FP} = 87\%$$

The second performance indicator that was used was the cellular mobile network type used by the smart phone with the Victim App during the System Test. The various network type used by mobile devices are 2G, 3G and 4G networks. From Table 4.1, all the four android smart phones have 2G and 3G network. Only Nokia 5(Victim App) and the INFINIX X559C had 4G networks.

Due to the unavailability of the 4G networks in most android smart phones and the smaller range it covers in comparison to both 2G and 3G network, the 2G and 3G networks were used to obtain real-time delivery of the SMS to the phones with the agency Apps. Variations where observed in the delivery time and was recorded.

Tables 4.5 and 4.6 shows the results obtained from 5 crashes each with 2G and 3G network respectively. Five crashes where simulated with the 2G network and another five with 3G network enabled on the Nokia 5 phone respectively, variations were observed in delivery time of the SMS received .

Table 4.5: Crash result showing SMS delivery time with 2G Networks.

Crash No.	Delivery Time in Seconds
1	8.7
2	7.2

3	7.4
4	5.6
5	5.8

Figure 4.7 shows the effect of the time variation recorded on a graph for the 2G network. The graph pattern is not as smooth as the values obtained and showed on figure 4.8 which is for 3G network. It took a longer time to deliver the SMS with the 2G network in comparison with the 3G network. This as a result, shows why the graph in figure 4.8 was smoother. The vertical axis represent the delivery time in seconds while the horizontal axis showed the number of crashes recorded.

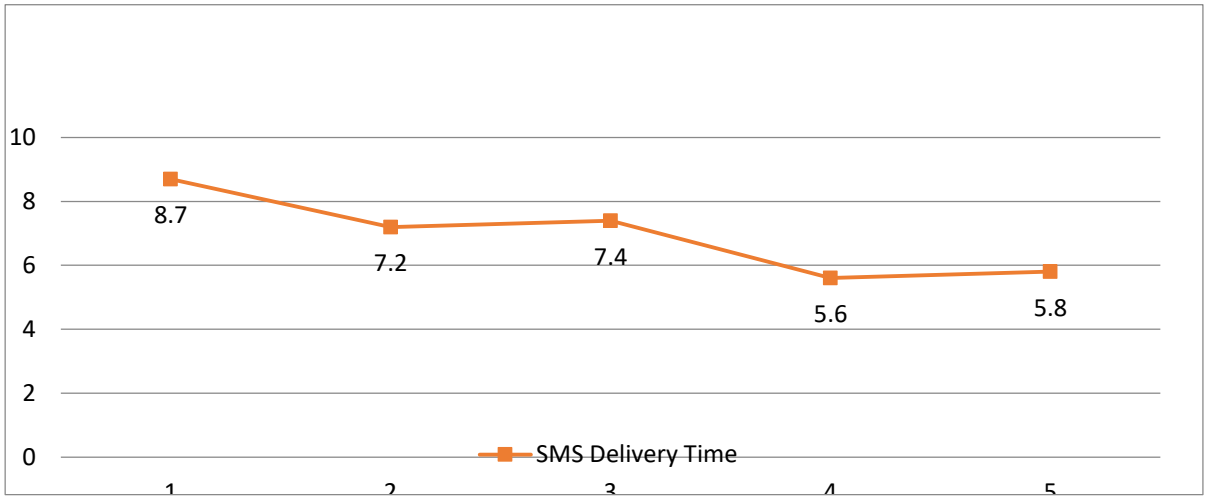


Fig. 4.9: SMS real-time delivery for 2G

Table 4.6: Crash result showing SMS delivery time with 3G Networks.

Crash No.	Delivery Time in Seconds

1	4.6
2	4.2
3	3.9
4	3.5
5	4.2

Figure 4.8 in the other hand represent a smoother steep, the delivery time by the 3G network was better than that of the 2G network.

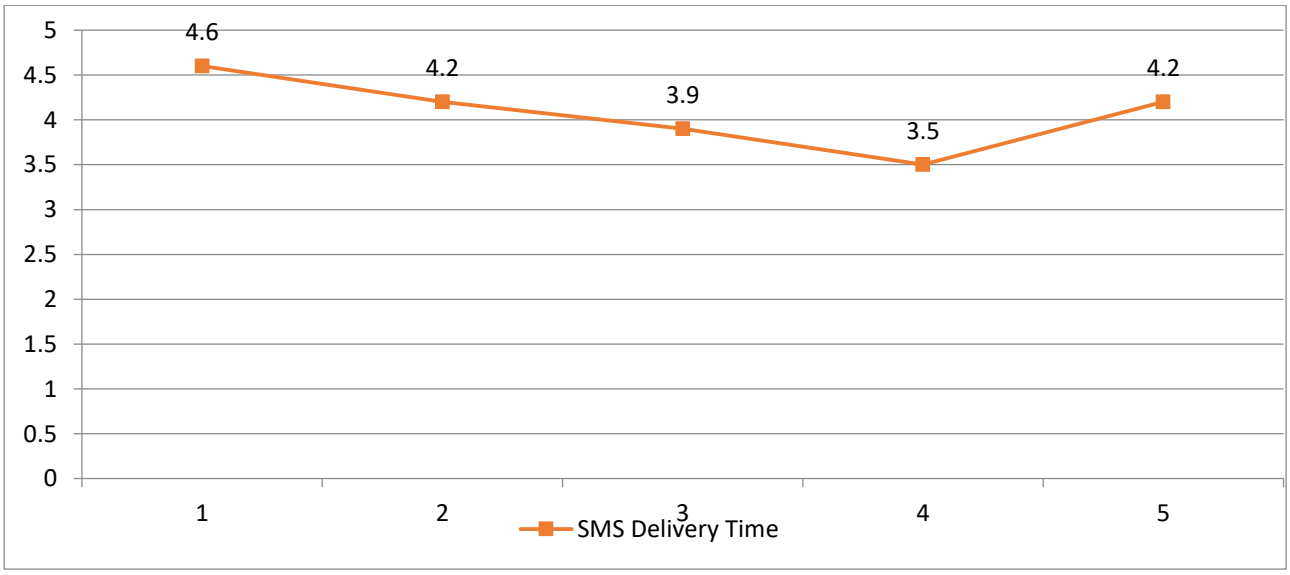


Fig. 4.10: SMS real-time delivery for 3G

Standard SMS rate was charged by the network service provider from the mobile phone number inside the victim's smart phone.

CHAPTER FIVE

5.0 CONCLUSION AND RECOMMENDATION

5.1 Conclusion

A mobile phone-based vehicle crash location detection and alerting system using android smart phone and GPS technologies was developed. The system was successfully implemented as both the Apps were successfully installed on different Android phones and each successfully retrieved the devices location using the devices inbuilt GPS triggered by its accelerometer (i.e., devices change in acceleration) and SMS was sent to the agency closest to the crash site successfully. This was achieved by the algorithm developed and integrated in the Victim's App for the system. The evaluation of the system was carried out and results of experiments were presented using a

confusion matrix and the cellular network type. The confusion matrix results obtained showed high Sensitivity, adequate accuracy and precision.

5.2 Recommendation

Two recommendations were made at the end of the research work which was:

i. Since the developed system used static mode of addressing for the registration of the three agencies used. The location where an agency was stationed during its registration is that same locations coordinates the algorithm fetched and perform the mapping operation. Using dynamic addressing will make the system more flexible where an already registered agency can change location without re-registering again. It is recommended that dynamic addressing should be used because that will make the agency to move around like ambulances and Red Cross vehicles do.

ii. Secondly, the network type signifies how quick the victim App retrieved the required information from the Google Firebase that was sent in the SMS. Due to the unavailability of 4G network, only 2G and 3G were used for the system. Since 4G is a faster network, it is recommended that it should be used.

5.3 Contribution to knowledge

The system designed was not only able to detect crash, it was able to send SMS to a first responder that is closest to the crash site. This will make it possible for the victim's that needs urgent help get prompt attention. The ability of the designed system to locate the closest emergency center from the crash site was the major contribution to knowledge.

5.4 Suggestion for Future Work

For future work, the addresses of the emergency centers can be made dynamic, this will allow the emergency centers that are mobile i.e. ambulances, to still receive the crash alert even when there

are moving around and 4G or 5G cellular network type should be used for the system in other to reduce the SMS delivery time of the system.

REFERENCES

Aloul, F., Zualkernan, I., Abu-Salma, R., Al-Ali, H. & Al-Merri, M. (2014). iBump: Smartphone application to detect car accidents, *IEEE International Conference on Industrial Automation, Information and Communications Technology (IAICT)*. 12(2), 52-56.

Azeez, R.A., Ogunrinde, M. A., & Sakirullah, O. A. (2015). A Web-based accident reporting and tracking system using sensor technology. *International Journal of Advances in Engineering and Technology*. 8(5), 678-684

Baramy, N., Singh, K. T., Jadhav, A., Javir, A. & Tarleka, S. (2016). Accident detection & alerting system, *International Journal of Technical Research and Applications.* 39, 8-11.

Dang, T.T., Truong, H., Tran, K.D (2016). Automatic fall detection using smart phone acceleration sensor. *(IJACSA) International Journal of Advanced Computer Science and Applications.*7(12), 12-17

Documentation for Firebase Cloud System. Retrieved from https://firebase.google.com/docs/cloud-messaging-2019-6

GB Blog Official, (2018). Retrieved from https://www.gearbest.com/blog/how-to/gps-and-gprs-tracking-2019-8

Goh, K.N., Jaafar, J., Mustapha, E.E.,Goh, E.T (2014). Automatic accident location detection System. *4th World Congress on Information and communication.* 2(8), 7-9

DOI:10.1109/WICT.2014.7077303

I.O.P (2014). How GPS works. Retrieved from http://www.physics.org/article-questions.asp?id=2019-10

Kaladevi, P., Kokila, T., Narmatha, S., & Janan, V. (2014). Accident Detection Using Android Smart Phone. *International Journal of Innovative Research in Computer And Communications Engineering*. 2(1), 2367-2372.

Khalil, U., Nasir, A., Javid, T., Raza, S. A., & Siddiqui, A. (2018). Automatic road accident detection using ultrasonic sensor. *International Symposium on Wireless Systems and Networks*. 5, 1-6, DOI: 10.1109/INMIC.2018.8595541

Narendar, D.S & Teja, R. (2013).Vehicle speed limit alerting and crash detection system at various zones. *International Journal of Latest Trends in Engineering and Technology*. 2(1), 108-113.

Njuguna, P. N. (2012). Instant GPS based motor vehicle accident detection and reporting. International Journal of I.C.T. *University of Nairobi School of Computing and Informatics*, 4, 12-15

Tanko, A. L. (2014). Evolution of mobile telecommunication with respect to 1G 2G, 3G and 4G. Retrieved from https://www.ncc.gov.ng/thecommunicator/2019-6

White, J., Thompson, C., Turner, H., Dougherty, B., & Schmidt, D.C. (2011). Wreckwatch: Automatic traffic accident detection and notification with smart phones. *Mobile Networks and Applications*, 16(3), 285-303.

WHO, "World Report on Road Traffic Injury Prevention: Summary" (2012). *World Health Organization*, Geneva, Switzerland.

Yogesh , J. I. & Rane, U.A. (2014). A Review on ARM7 based accident detection using GSM, GPS and MEMS. International Journal of Engineering Sciences & Research Technology. *Department of Electronics and Telecommunication Engineering , Shegaon, Maharashtr SSGMCE*

Zahradnik, F. (2017). An overview on how GPS technology works. Retrieved from https://www.lifewire.com/more-2018-7

APPENDIX A (SOURCE CODES)

```
package com.example.codesigood.crashdetector;

import android.support.annotation.NonNull;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.support.v4.view.GravityCompat;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;

public class DashboardActivity extends AppCompatActivity  {

    private final int MY_PERMISSION_REQUEST_CODE = 1;
    private PermissionHandler mPermissionHandler;
```

```
    private FirebaseAuth firebaseAuth;
    private FirebaseUser firebaseUser;
    private ServiceHandler mServiceHandler;
    private static boolean isTracking;
    private Button buttonToggleTracking;
    private DBEmergency mDatabase;
    private LocationManager locationManager;
    private DrawerLayout mDrawerLayout;
    private ActionBarDrawerToggle mDrawerToggle;
    private CharSequence mDrawerTitle;
    private CharSequence mTitle;
    ExpandableListAdapter listAdapter;
    ExpandableListView mDrawerexpList;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_dashboard);

        locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
       // custom_font = Typeface.createFromAsset(getAssets(), "AvenirNextLTPro-
MediumCn.otf");
        mPermissionHandler = new PermissionHandler(this);
        firebaseAuth = FirebaseAuth.getInstance();
        mServiceHandler = new ServiceHandler(this);
        isTracking = false;
        buttonToggleTracking = (Button) findViewById(R.id.buttonToggleTracking);
        CustomToastActivity.CustomToastActivity(this);
        mDatabase = new DBEmergency(this);

        setupFirebase();

        prepareListDataSignin();
        listAdapter=new
ExpandableListAdapter1(this,listDataHeader,login_icons,custom_font,custom_font);

        mTitle = mDrawerTitle = getTitle();
        mDrawerexpList = (ExpandableListView)findViewById(R.id.left_drawer);
        mDrawerexpList.setGroupIndicator(null);
      //  custom_font = Typeface.createFromAsset(getAssets(), "AvenirNextLTPro-
MediumCn.otf");
        mDrawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);

        // set a custom shadow that overlays the main content when the drawer opens
        mDrawerLayout.setDrawerShadow(R.drawable.drawer_shadow, GravityCompat.START);
```

```
 //set drawer expandable list adapter
      mDrawerexpList.setAdapter(listAdapter);
 //Creation of expandable listView
      mDrawerexpList.setOnGroupClickListener(new
ExpandableListView.OnGroupClickListener() {

         @Override
         public boolean onGroupClick(ExpandableListView parent, View v,
                              int groupPosition, long id) {
            if (groupPosition == 0) {
               // Already in this Activity
               mDrawerLayout.closeDrawer(mDrawerexpList);
            }
            if (groupPosition == 1) {
               finish();
               Intent intent=new Intent(DashboardActivity.this, MyEmerContActivity.class);
               startActivity(intent);
               mDrawerLayout.closeDrawer(mDrawerexpList);
            }
            if (groupPosition == 2) {
               finish();
               Intent intent=new Intent(DashboardActivity.this, MyAccount.class);
               startActivity(intent);
               mDrawerLayout.closeDrawer(mDrawerexpList);
            }
            if (groupPosition == 3){
               logout();
            }
            return false;
         }
      });
      // Listview Group expanded listener
      mDrawerexpList.setOnGroupExpandListener(new
ExpandableListView.OnGroupExpandListener() {

         @Override
         public void onGroupExpand(int groupPosition) {
         }
      });

      // Listview Group collasped listener
      mDrawerexpList.setOnGroupCollapseListener(new
ExpandableListView.OnGroupCollapseListener() {

         @Override
         public void onGroupCollapse(int groupPosition) {
```

```
        }
    });

    final android.support.v7.widget.Toolbar toolbar = (android.support.v7.widget.Toolbar) findViewById(R.id.toolbar);
    mDrawerToggle = new ActionBarDrawerToggle(

private void setupFirebase() {
    firebaseAuth = FirebaseAuth.getInstance();
    firebaseUser = firebaseAuth.getCurrentUser();
    if (firebaseUser == null) {
        finish();
        startActivity(new Intent(this, LoginScreenActivity.class));
    }
}

public void logout()
{
    try {
        firebaseAuth.signOut();
        finish();
        startActivity(new Intent(this, LoginScreenActivity.class));
    } catch (Exception e) {
        Toast.makeText(this, "Unsuccessful", Toast.LENGTH_SHORT).show();
        e.printStackTrace();
    }
}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    switch (requestCode) {
        case MY_PERMISSION_REQUEST_CODE:
            if (mPermissionHandler.handleRequestResult(requestCode, permissions, grantResults))
{
                toggleTracking();
            }
            break;
        default:
            super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    }
}

private boolean locationServicesStatusCheck() {
```

```
    final LocationManager locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);

    if (locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER)) return true;

    AlertDialog.Builder builder = new AlertDialog.Builder(DashboardActivity.this);
    builder.setTitle("Enable GPS")
            .setMessage("This function needs your GPS, do you want to enable it now?")
            .setIcon(android.R.drawable.ic_menu_mylocation)
            .setCancelable(false)
            .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialogInterface, int i) {
                    startActivity(new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS));
                }
            });
    AlertDialog dialog = builder.create();
    dialog.show();


    return false;
  }

  private boolean hasContact() {
    String email = firebaseUser.getEmail();
    List<EmerContact> contact = mDatabase.getContact(email);
    if (contact.isEmpty()) {
      CustomToastActivity.showCustomToast("Please add at least 1 Emergency Contact");
      return false;
    } else {
      return true;
    }
  }

  private void prepareListDataSignin() {
    listDataHeader =new ArrayList<String>();
    //listDataChild = new HashMap<String, List<String>>();

    // Adding group data
    listDataHeader.add("Dashboard");
   // listDataHeader.add("Emergency Contacts");
    listDataHeader.add("My Account");
    listDataHeader.add("Log Out");

  }
```

//SendSMS Activity

```
package com.example.codesigood.crashdetector;

import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.support.v4.view.GravityCompat;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support.v7.app.AlertDialog;
import                                          android.support.v7.app.AppCompatActivity;import
android.widget.ExpandableListAdapter;
import android.widget.ExpandableListView;
import android.widget.TextView;
import android.widget.Toast;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

public class DashboardActivity extends AppCompatActivity  {

    private final int MY_PERMISSION_REQUEST_CODE = 1;
    private PermissionHandler mPermissionHandler;

    private FirebaseAuth firebaseAuth;
    private FirebaseUser firebaseUser;
    private ServiceHandler mServiceHandler;
    private static boolean isTracking;
    private Button buttonToggleTracking;
    private DBEmergency mDatabase;
    private LocationManager locationManager;
    private DrawerLayout mDrawerLayout;
    mPermissionHandler = new PermissionHandler(this);
        firebaseAuth = FirebaseAuth.getInstance();
        mServiceHandler = new ServiceHandler(this);
        isTracking = false;
        buttonToggleTracking = (Button) findViewById(R.id.buttonToggleTracking);
        CustomToastActivity.CustomToastActivity(this);
        mDatabase = new DBEmergency(this);
  setupFirebase();
prepareListDataSignin();
        listAdapter=new
ExpandableListAdapter1(this,listDataHeader,login_icons,custom_font,custom_font);
mTitle = mDrawerTitle = getTitle();
```

```
        mDrawerexpList = (ExpandableListView)findViewById(R.id.left_drawer);
        mDrawerexpList.setGroupIndicator(null);
      //          custom_font          =          Typeface.createFromAsset(getAssets(),          "AvenirNextLTPro-
MediumCn.otf");
        mDrawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);

        // set a custom shadow that overlays the main content when the drawer opens
        mDrawerLayout.setDrawerShadow(R.drawable.drawer_shadow, GravityCompat.START);

    private void setupFirebase() {
        firebaseAuth = FirebaseAuth.getInstance();
        firebaseUser = firebaseAuth.getCurrentUser();
        if (firebaseUser == null) {
            finish();
            startActivity(new Intent(this, LoginScreenActivity.class));
        }
    }

    public void logout()
    {
        try {
            firebaseAuth.signOut();
            finish();
            startActivity(new Intent(this, LoginScreenActivity.class));
        } catch (Exception e) {
            Toast.makeText(this, "Unsuccessful", Toast.LENGTH_SHORT).show();
            e.printStackTrace();
        }
    }


Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS));
                    }
                });
        AlertDialog dialog = builder.create();
        dialog.show();
        return false;
    }

    private boolean hasContact() {
        String email = firebaseUser.getEmail();
        List<EmerContact> contact = mDatabase.getContact(email);
        if (contact.isEmpty()) {
            CustomToastActivity.showCustomToast("Please add at least 1 Emergency Contact");
            return false;
        } else {
```

```
            return true;
        }
    }

    private void prepareListDataSignin() {
        listDataHeader =new ArrayList<String>();
        //listDataChild = new HashMap<String, List<String>>();

        // Adding group data
        listDataHeader.add("Dashboard");
       // listDataHeader.add("Emergency Contacts");
        listDataHeader.add("My Account");
        listDataHeader.add("Log Out");

    }


    public boolean onCreateOptionsMenu(Menu menu) {
        return super.onCreateOptionsMenu(menu);
    }
```

// SensorActivity

```
package com.example.codesigood.crashdetector;
public class SensorService extends Service implements SensorEventListener

    // TAG to identify notification
    private static final int NOTIFICATION = 007;
 // IBinder object to allow Activity to connect
    private final IBinder mBinder = new LocalBinder();

    // Sensor Objects
    private Sensor accelerometer;
    private SensorManager mSensorManager;

    private double accelerationX, accelerationY, accelerationZ;

    private int threshold = 15;

    // Notification Manager
    private NotificationManager mNotificationManager;

    @Override
    public IBinder onBind(Intent intent) {
        // TODO: Return the communication channel to the service.
```

```
        return mBinder;
    }

    @Override
    public boolean onUnbind(Intent intent) {

        return super.onUnbind(intent);
    }

    public class LocalBinder extends Binder {
        public SensorService getService() {
            return SensorService.this;
        }
    }

    public void onSensorChanged(SensorEvent sensorEvent) {
        accelerationX = (Math.round(sensorEvent.values[0]*1000)/1000.0);
        accelerationY = (Math.round(sensorEvent.values[1]*1000)/1000.0);
        accelerationZ = (Math.round(sensorEvent.values[2]*1000)/1000.0);
        /*** Detect Accident ***/
        if (accelerationX > threshold || accelerationY > threshold || accelerationZ > threshold) {
            Intent mIntent = new Intent();
            mIntent.setClass(this, SendSMSActivity.class);
            mIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            mSensorManager.unregisterListener(this);                    // Unregister sensor when not
in use
            mNotificationManager.cancel(NOTIFICATION);
            stopSelf();
            startActivity(mIntent);
```

AGENCY

//Agent

```
package com.example.codesigood.agency;

public class Agents {

    private String name,phone,userId;
    private double lat,lot;

public Agents(String name, String phone,String userId,double lat,double lot)
    {
        this.name = name;
        this.phone = phone;
        this.userId = userId;
```

```
        this.lat = lat;
        this.lot =lot;
    }
    public String getName() {
        return name;
    }

    public String getPhone(){return phone;}
    public String getUserId(){return userId;}
    public double getLat(){return lat;}

    public double getLot() {
        return lot;
    }
}
```

// Register

```
package com.example.codesigood.agency;
import org.w3c.dom.Text;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.TimeZone;

public class Register extends AppCompatActivity {
    private final int LOCATION_REQUEST_CODE = 199;
    EditText fullname,email,password,phone;
    private ProgressDialog progressDialog;
    private FirebaseDatabase fireDatabase;
    private FirebaseAuth fireAuth;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);
        initial();
        isTracking = false;
        mPermissionHandler = new PermissionHandler(this);
        startTrackingWrapper();
        locationUpdater();
    }
    public void initial(){
```

```
        phone = (EditText) findViewById(R.id.phone);
        fullname = (EditText)findViewById(R.id.regName);
        email = (EditText) findViewById(R.id.regEmail);
        password = findViewById(R.id.regPassword);
        progressDialog = new ProgressDialog(this);
        fireAuth =FirebaseAuth.getInstance();
    }

   public void register(View view){
       final String name = fullname.getText().toString().trim();
        final String emails = email.getText().toString().trim();
       final String pass = password.getText().toString().trim();
       final String phones =phone.getText().toString().trim();
        if(!Patterns.EMAIL_ADDRESS.matcher(emails).matches())
        {
           Toast.makeText(getApplicationContext(),"Sorry!                                    invalid
email",Toast.LENGTH_LONG).show();
           return;
        }
        if(TextUtils.isEmpty(emails)){
              return;
        }
        if(TextUtils.isEmpty(name))
        {
           return;
        }
        if(TextUtils.isEmpty(pass))
        {
           return;
        }

           progressDialog.setMessage("Registering....  Agent!");
           progressDialog.show();
locationUpdater();
       fireAuth.createUserWithEmailAndPassword(emails,pass)
                .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
              @Override
              public void onComplete(@NonNull Task<AuthResult> task) {
                 if(task.isSuccessful())
                 {
                    fireAuth.signInWithEmailAndPassword(emails,pass);
                    FirebaseUser user  = fireAuth.getCurrentUser();
                    dataReference          =          FirebaseDatabase.getInstance().getReference("Agent
Informations");
                    String key = user.getUid();
```

```
                Agents agents = new Agents(name,phones,key,mCurrentLocation.getLatitude(),mCurrentLocation.getLongitude());
                dataReference.child(key).setValue(agents);
                // saveToFirebase();
              progressDialog.dismiss();
                Toast.makeText(getApplicationContext(),"Registration Successfull !",Toast.LENGTH_LONG).show();
                startActivity(new Intent(getApplicationContext(), MainActivity.class));
                finish();
                //finish();

              }else{

                Toast.makeText(getApplicationContext(),"Faild to register Agent!",Toast.LENGTH_LONG).show();
              }

            }
        });

  }
   public void locationUpdater()
   {
      locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
      if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
         return;
      }
      if (locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER)) {
         //Toast.makeText(getApplicationContext(), "Tnx Network", Toast.LENGTH_LONG).show();
         locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 0, 2, new LocationListener() {
            @Override
            public void onLocationChanged(Location location) {
               Toast.makeText(getApplicationContext(), "Tnx for updating..",
            }

            @Override
            public void onStatusChanged(String provider, int status, Bundle extras) {
               locationManager.removeUpdates(this);
            }
```

```
        @Override
        public void onProviderEnabled(String provider) {

        }

        @Override
        public void onProviderDisabled(String provider) {

        }
    });
```

// JSON-Firebase

```
package com.example.codesigood.agency;

import android.util.Log;
import com.google.android.gms.maps.model.LatLng;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

/**
 * Created by anupamchugh on 27/11/15.
 */

public class DirectionsJSONParser {
   /** Receives a JSONObject and returns a list of lists containing latitude and longitude */
   public List<List<HashMap<String,String>>> parse(JSONObject jObject){

      List<List<HashMap<String, String>>> routes = new
ArrayList<List<HashMap<String,String>>>() ;
      JSONArray jRoutes = null;
      JSONArray jLegs = null;
      JSONArray jSteps = null;

      try {

         jRoutes = jObject.getJSONArray("routes");

         /** Traversing all routes */
         for(int i=0;i<jRoutes.length();i++){
            jLegs = ( (JSONObject)jRoutes.get(i)).getJSONArray("legs");
            List path = new ArrayList<HashMap<String, String>>();
```

```
 * Method to decode polyline points
    * Courtesy : http://jeffreysambells.com/2010/05/27/decoding-polylines-from-google-maps-direction-api-with-java
    * */
   private List decodePoly(String encoded) {

       List poly = new ArrayList();
       int index = 0, len = encoded.length();
       int lat = 0, lng = 0;

       while (index < len) {
          int b, shift = 0, result = 0;
          do {
             b = encoded.charAt(index++) - 63;
             result |= (b & 0x1f) << shift;
             shift += 5;
          } while (b >= 0x20);
          int dlat = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
          lat += dlat;

          shift = 0;
          result = 0;
          do {
             b = encoded.charAt(index++) - 63;
             result |= (b & 0x1f) << shift;
             shift += 5;
          } while (b >= 0x20);
          int dlng = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
          lng += dlng;

          LatLng p = new LatLng((((double) lat / 1E5)),
                (((double) lng / 1E5)));
          poly.add(p);
       }

       return poly;
   }
}
```

CHAPTER ONE

1.0 INTRODUCTION

1.1 Background of the Study

Accidents caused by vehicular traffics results in a lot of human injuries and death. Delay in dispatching emergency aids to accident location is one of the most important indicators for survival rate. A vehicle equipped with a system that can automatically detects accident and quickly alert first responders is one of the ways used to eliminate such delays, (White *et al*., 2011)**.** The majority of vehicles using our roads do not have this automatic detection system and it is expensive to retrofit for older vehicles.

Furthermore, traffic is on the rise as the demand for vehicles is getting higher, so the means of transportation needs to be improved upon. Accident victims need to get help on time to reduce fatality rate. Figure 1.1 shows a crashed vehicle with an in-built accident detection system. The crash activates auto dialer inside the car and sends GPS location and the vehicle registration number to a first responder, an insurance company if the vehicle has insurance and to family members.

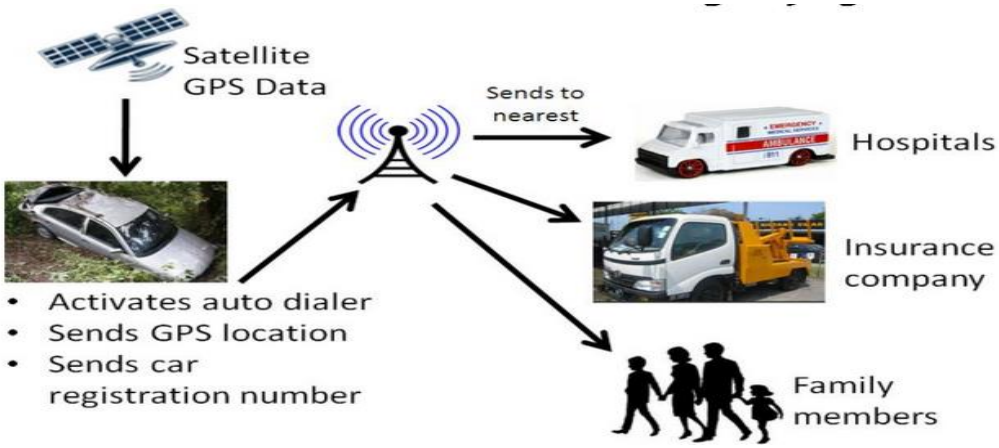


Fig.1.1 Automatic Accident Location Detection System (AALDS)
(Goh *et al.*, 2014)

However, various accident detection systems have been proposed with different methodologies. One of such methods is the use of the Arduino micro controller with other components like the GPS receiver and GSM/GPRS components in other to detect and send location of an accident for quick response of first responders. Real-time control of vehicle speed automatically could be very difficult to realize, instead of controlling the vehicle speed automatically, a system that detects accidents and also keep the driver of a vehicle on alert about the speed limit was proposed. The

system can detects when a driver of a vehicle exceeds the speed limits and ignores the warning. Using the GSM technology incorporated in the system design, an SMS will be sent to a patrol corps with the location of the speeding vehicle, (Narendar and Teja, 2013).

In a world that is becoming more motorized, impact sensors in conjunction with GPS offers a timely, objective and potentially acceptable method of detecting and reporting accidents, (Njuguna, 2012).

In addition, smart phones are cheap and easy to purchase in comparison with other traffic accident prediction system, this makes them a very good alternative to use. Moreover, even if a car is not fitted with an accident detection and notification system, smart phones generally travel with their owners to detect accidents, (White *et al*., 2011).

1.2 Statement of the Problem

When a vehicular accident occurs, it is usually the good Samaritans that report such accidents to the emergency responders. The moment between when an accident happens to the time where such emergency centers are notified is a key indicator to saving lives. Waiting for good Samaritans to report accident will lead to wasting precious time which the victims may not have. Having an automatic notification system that can notify the emergency center that is closest to the accident site will go a long way in reducing the level of causalities. In addition, the various in-built car crash detection and alerting systems are mostly available in luxury cars. These cars are mostly saloon and have maximum sitting capacity from 5-9 people commuting per vehicle. The majority of vehicles on our roads don't have this facility. Android smart phones are one of the cheapest smart phones available and it is believed that at least one in every five people possesses it. Utilizing

these facilities in android smart phone to develop a system that can detect accidents will reduce the fatality rate of accidents.

1.3 Aim and Objectives of the Research Study

The aim of this research work is to develop a vehicle crash location detection and alerting system using android smart phone and GPS technologies.

The specific objectives of the study were:

iv. To set an acceleration threshold for a smart phone that can detect crash using the smart phone accelerometer and GPS technologies.
v. To develop an algorithm that maps an accident location to an emergency center closest to the accident location and send notification to that center.
vi. To evaluate the performance of the developed system using accuracy, precision, speed, sensitivity and cellular network type as the performance metrics.

1.4 The Significance of the Study

The research work will eliminate the delay in notifying an accident to first responders such as the Nigerian Road Safety Cooperation and Red Cross. Automatic notification using the android smart phone and the GPS technologies will provide timely and accurate information to these organizations and eliminate delay in time between the occurrence of an accident to when such accident is reported. This will go a long way in saving more lives of accident victims because the required aid and attention will be brought to the accident site immediately.

1.5 Scope of the Study

In the proposed system, only android smart phones were chose and used for the system test. Even though there are other smart phones, the availability and the ease of use of the android technology prompted the decision. A remote controlled toy car was used to simulate the vehicle that will be involved in an accident; this is because using a real vehicle will be expensive to manage.

1.6 Limitation of the Research Study

The initial idea was to design a system that would be able to use dynamic form of addressing to store the addresses of the agencies on a Google cloud firebase. But due to financial constrain involved in securing full functionality of the firebase from Google and the complications in making the various components of the system to communicate, static addressing was later adopted. Dynamic addressing would allow the emergency centers to move around and still receive notification of an accident based on their distances from the accident location. In addition, the system was tested with a radio controlled toy car which cannot give a perfect simulation of accident.

1.7 Definition of Terms

Alert System: A system that can send notification out to targeted persons or devices when it is triggered.

Global Positioning System (GPS): A satellite and radio navigation based system that is owned and regulated exclusively by USA Air force.

Accelerometer: Is an instrument that is used to measure moving or vibrating body acceleration in all directions.

Smart Phone: A mobile phone that performs like a computer in many different ways, typically, having internet access, a moderate to large touch screen and an operating system that can run applications that are downloaded from different sources.

Firebase: Is a cloud-hosted NoSQL real-time database that enables information to be stored and synchronized in real-time.

Mobile Application (App): These are developed computer programs or software application that are designed and are capable of running on devices that are mobile such as phone.

False Negative: An instance where the mobile-phone does not detect a crash even though a crash has occurred

False Positive: An instance where the mobile phone detects a crash even though it has not occurred

True Positive: Number of actual crash instances that the system classified as crashes

True Negative: Number of non-crash instances that the system classified as false crash

CHAPTER TWO

2.0 LITERATURE REVIEW

2.1 Review of Some Related Literature

Crashes especially those involving loss of lives around the world and in particular developing countries is alarming, (Azeez *et al*., 2015). In 2014, W.H.O reported that, even though road crashes are sometimes avoidable, vehicle road accidents caused the undesirable deaths of 1.2 million people worldwide each year and injures about 4 times this number. There are formal and informal ways of reporting road crashes. Most often, good Samaritans report road crashes formally to the concerned agencies, through various channels, which include physical reporting to the road traffic office or telephone call to the road traffic emergency numbers. Studies have shown that delay in reporting a road crash to the appropriate agency and the time it takes the rescue agencies to arrive at the accident location is primarily the biggest reason why causality rate is high in serious road crashes, (Aloul *et al*., 2014).

Casual inquiries from the concerned agencies indicate slow and cumbersome methods of crash detection, reporting and locating the scene of the road crash. For this purpose, some vehicle manufacturing companies embed accident detection systems in their vehicles. These systems use impact sensors to generate impact signals that are fed into Geographical Positioning System (GPS) receivers, which communicate with GPS satellites. In this information age, almost every individual owns a mobile phone; especially smart phones. Smart phones can be found everywhere and with network connectivity, its in-built facilities can be utilized to develop a lot of user applications. Recently released models of smart phones support sensors including audio recorders, GPS and accelerometers in addition to much other functionality, (Aloul *et al.*, 2014).

Most smart phones have wireless information facilities that provide extra possibilities to build consumer apps that take advantage of the sensors and network connectivity that the different kinds of. Therefore, impact signals generated using in-phone accelerometer and GPS can be used to provide real-time and adequate method of reporting and locating road crash scene. With the location tracking capabilities of GPS device, it becomes easier to generate location coordinates of a crashed vehicle.

Yogesh and Rane (2014) reviewed the various accidents detection systems and proposed an embedded system. The system proposed was Advanced RISC Machines (ARM) based accident detection using GSM, GPS and Micro Electro-Mechanical Sensor (MEMS). Some steps that will be helpful in reducing the rescue time were outlined. These steps are firstly, fast and accurate accident detection and reporting to a Public Safety Answering Point (PSAP). Secondly, fast and efficient evacuation of occupants trapped inside a vehicle. The proposed system used sensors like

opt coupler, ultrasonic, alcohol sensors and subsystems like MEM. The system is hardware based and the entire work has to be integrated with the automobile to validate its functionality and reliability which will be expensive to achieve.

Khalil *et al.* (2018) proposed automatic road accident detection using ultrasonic sensor that can detect when a car is close to an object which could be another car. The ultrasonic sensors are placed inside the car on two positions, the front windscreen and the back windscreen. Two thresholds were set for both of the sensors. As the care gets near an object, the distance between the two sensors decreases and this immediately set an alarm. GPS was used to find location of the car is. The system relied majorly on the sensors to avoid crash; however, sensors are not 100 percent reliable because they can fail any time.

Beramy *et al*. (2016) proposed accident detection and alerting system that detects road crashes, gets the location of the crash and sends an SMS containing the location to a hospital or police station. The system is a hardware built on a microcontroller board (Arduino board), with a GPS module and a GSM module attached to it. The system uses a 3-axis accelerometer to detect accidents. When an accident is detected the micro-controller gets the coordinates from the GPS and uses the GSM modem to send the notification message. The modules are first turned on, then the GSM module acquires network signal then the system goes into a standby mode. When an accident occurs, the impact sensor gives a positive output and the micro-controller acquires the coordinate's location using the GPS module. The coordinates are integrated into the notification message and then sent to emergency services and contacts predefined by the user. The system is hardware based; retrofitting the designed system for older vehicle will be very expensive,

Narendar and Tejar (2013) developed a system that can detect a vehicle crash at the same time keep the driver of the vehicle on a serious alert if the speed limit is exceeded, using Micro Electromagnetic System (MEMS) to detect road crashes, Radio-frequency Identification (RFID) to detect various zones, a GPS module to get location coordinates and a GSM module to send accident details and location and also the vehicle and speed limit details to traffic police was developed. The main idea was to come up with a controller that has a smart display that is meant for vehicles speed limit and crash alerts which can run on an embedded system. When an accident occurs, MEMS sensors begins the process, then the micro-controller triggers and activates the module of the GPS to get the location of the vehicle and also the GSM module is activated to send the SMS. The sensors and other components that are needed for the design are expensive to acquire.

(Njuguna, 2012) developed a vehicle accident detection and reporting solution that is an Instant GPS motor based. The system consisted of components such as accident detection component, vehicle tracking component and accident surveillance module; the accident detection module has impact sensors, signal conditioning electronics and accident detection logic unit. Active GPS device with three important capabilities including a GPS signal interface, digital input interface and a wireless interface with transmitting capabilities make up the vehicle tracking module. The surveillance module has a computer with an input interface for a wireless GSM enabled mobile phone. The computer hosts a database for recording accident incidents, accessible through a website, the recorded accident data is displayed at the client workstation against a GIS digital map backdrop, using customized software to authorized users of the system via the website and the

internet technology. New accident incident will sound an alarm at the client side to alert the user. This enables authorized users, for example the police and ambulance paramedics to learn about an accident occurrence, allowing them to track and move to the accident site for evacuations. The system relied on too many components like the website, digital interface, and signal conditioning electronics to function.

Dang *et al*. (2016) designed automatic fall detection system using smart phone acceleration sensor that works with the accelerometer of the smart phone. A threshold was set for the accelerometer. The system was meant to detect if there is an accidental fall that will be strong enough to trigger the acceleration threshold that was set. The system was tested and was meant for old persons or people with ailment that needed to be monitored constantly to avoid dangerous fall. The smart phone will be attached to a person's body, in an eventual fall, an alarm will be triggered and the Smartphone will make sound for as long as the alarm clock is set. The system is only designed to detect a fall; it is not designed to notify emergency responders in a situation where the fall is fatal.

Kaladevi *et al.* (2014) developed and implemented an accident detection system using an android smart phone. This system uses the heart beat sensor (IR sensor and microphone) attached to the seat belt to measure the heart beat rate, the measured data is then transmitted to the phone using Bluetooth. This is achieved via the control unit (micro-controller) which is connected to the heart beat sensor and is also installed in the vehicle. When there is variation from the normal heart beat rate, then an accident is detected. The android smart phone then uses its inbuilt GPS to get the crash location and then send the message to an emergency contact. However, heart beat rate is not entirely reliable because it cannot be controlled in certain circumstances.

Accelerometers and acoustic data when properly used by smart phones can detect traffic accident and send notification and situational awareness via coordinates from GPS to a central server created for emergency personnel. The system developed is WreckWatch; it was built for android smart phones. It has two phases a client side installed on a smart phone which detects accident, show map and gives provision for registering emergency contacts, and also gives room for eyewitness report to provide situation awareness data, and a server side this offers aggregation of information and a means of communication for emergency staff, family and friends. It also enables customers to submit accident features (such as acceleration, path, and velocity) and provides multiple interfaces, such as any Google map and Extended Mark-Up Language/JavaScript Object Notation (XML/JSON) services for web and for accessing the provided information, (White *et al.,* 2011).

2.1.1 Android Smart Phones

These are the smart phones that run on the android operating system. They can perform the typical computer functions, access the internet and have a moderate to large touch screen. The operating system is capable of running downloaded Apps. Smart phones has number Sensors that your software can use, such as accelerometer, magnetometers, proximity sensors, barometer and gyroscope and support wireless communications protocols such as Bluetooth, Wi-Fi, and satellite navigation, (http://www.wikipeadia.com).

2.1.1.1 Some Key Features of Android Smart Phones

iv. Central Processing Unit (CPU)

The CPU of smart phones is optimized to perform typical functions of computer. The output of the portable CPU relies on what you want, not just on the clock rate but also on the memory hierarchy. They can effectively perform in a low power environment, (http://www.wikipeadia.com).

v. Display

The screen is one of the main features of android smart phone. Depending on the device design, the screen fills most or nearly all of the space on a device front service, (http://www.wikipeadia.com).

vi. Battery

The battery life of smart phone has gradually been improved upon depending on the model of the phone. Earlier smart phones battery could not handle significantly the power requirements of the embedded processor and color screens.

iv. Location detection Ability

Smart Phone has the ability to easily detect your location, and even keep track of your movements. This location detection is achieved with the inbuilt GPS technology on the smart phone, (http://www.wikipeadia.com).

2.1.2 How GPS is working

GPS with the help of satellites transmits very accurate signals, enabling GPS receivers to calculate and show the user's exact location, velocity and time data. GPS receivers can use the mathematical principle of trilateration to determine your location by capturing the signals from satellites. With the addition of computing power and data stored in memory such as road maps, points of interest,

topographic information, and much more, GPS receivers can convert data about place, velocity and time to a helpful display form, (Zahradnik, 2017).

GPS is a network of approximately 30 Earth orbiting satellites at 20,000 km altitude. The system was initially created by the U.S. government for military navigation, but now anyone with a GPS device, whether it is a SatNav, a mobile phone or a handheld GPS unit, can receive the satellite radio signals.

Wherever you are on the planet, there are' noticeable' at any moment at least four GPS satellites. Each transmits data at periodic intervals about their place and the present moment. Your GPS receiver intercepts these signals, traveling at the velocity of light, which calculates how far each satellite is based on how long it took for the messages to arrive. Once you have data about how to do it, Institute of Physics (I.O.P, 2014).

2.1.2.1 GPS Trilateration

Imagine you are standing somewhere on Earth with three satellites in the sky above you. Figure 2.1 shows GPS trilateation. If you understand how far you're from satellite A, you know you have to be on the red circle somewhere. You can work out your place by seeing where the three circles intersect if you do the same for satellites B and C. While using overlapping spheres rather than circles, this is just what your GPS receiver does. The more satellites above the horizon, the more correctly they are, (I.O.P, 2014).

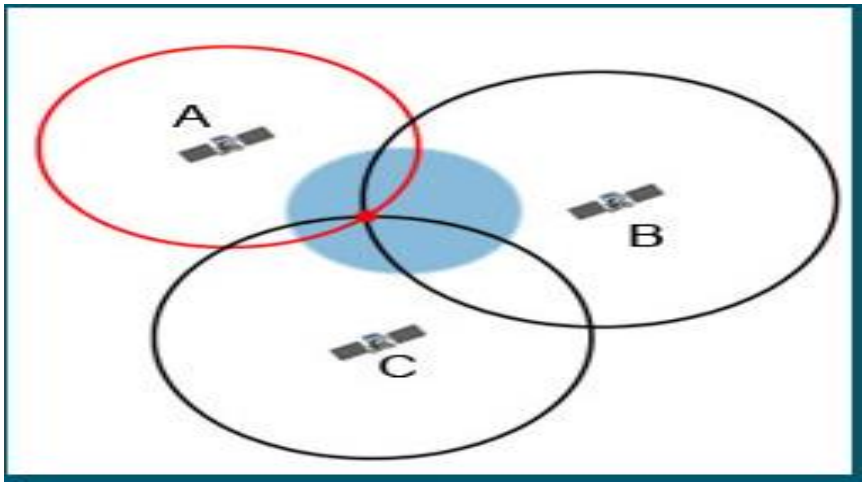


Fig.2.1: GPS Trilateration Illustration (Source: I.O.P, 2014).

2.1.2.2 Limitations of GPS

Barriers like thick forest, canyon walls, skyscrapers, bridges can block the GPS signal by making it hard or impossible to navigate accurately. Similarly, indoor and underground areas, GPS does not operate well. Maintenance of satellites, radio interference and solar storms can cause gaps in coverage.

2.1.3 Difference between GPS and GPRS

GPS stands for Global Positioning System whereas GPRS stands for General Packet Radio Service. In simple terms, GPS will give you the Latitude and Longitude coordinates location while GPRS will allow you to transfer data over cellular networks.

GPS technology can identify any Earth position or address. Although it involves the operation of various satellites, it can be used almost anywhere in the globe, making it highly available and

advantageous for companies with many distinct geographical places. GPRS is the wireless information service most frequently used. Older versions relied on 2G cellular networks, but now most use 3G technologies. GPRS enables cellular devices to perform functions such as multimedia messages and Internet surfing, (GB Blog Official, 2018).

2.2 Evolution of 2G, 3G and 4G Networks in Mobile Phones

Mobile devices need a strong and reliable cellular network in other to be able to effectively use the GPRS facility built on them. For the system to work, a good network type is needed based on some certain network parameters such as speed, bandwidths, frequency, and accessibility. All android smart phones have 2G and 3G. Basically, to use any of the network type, the owner of a smart phone must choose between the network types. 4G network have higher speed and bandwidth but are not supported by all the cellular network service providers,

2.2.1 Second Generation Mobile Telecommunication (2G)

2G networks started to appear around the late 1980s and in many nations these networks arrived in the early 1990s. In contrast to its predecessor which was the 1G network, Digital modulation methods were used in these technologies, resulting in superior speech quality. But the circuit stayed turned to the networks. 2G got fresh facilities like SMS, fax, and Web Appilication Protocol (WAP). Encryption was introduced, that greatly improved safety and solved most 1G security issues, as well as improved service quality for error detection and correction. Some 2G systems include the Global Mobile Communications System (GSM), IS-54 (digital AMPS), IS-95 (CDMA), GPRS and EDGE, (Tanko, 2014).

While 2 G technologies significantly enhanced mobile communications, leading to an explosion in numbers of subscribers, it was with many constraints. One of these includes the fact that 2G was a circuit-based network, so it utilizes bandwidth and resources inefficiently, which greatly limits the capacity of elevated information rates, is also unable to manage complicated information such as video and also limits the amount of customers. Other limitations include the lack of interoperability between 2G networks, poor standardization and the fact that 2 G provides very limited services and apps, (Tanko, 2014).

2.2.2 Third Generation Mobile Telecommunication (3G)

3G Mobile networks have been created based on the International Telecommunications Union (ITU), a unified family of norms capable of working together and meeting International Mobile Telecommunications (IMT-2000) specifications, to construct mobile networks that deliver multimedia services and other services accessible on wireless devices. These networks began operating mainly around the beginning of the 2000s. 3G technologies used circuit switching for voice/SMS and packet switching for data services. The technologies include Wideband-Code Division Multiple Access (W-CDMA), Code Division Multiple Access (CDMA-2000) and Time Division-Synchronous Code Division Multiple Access (TD-SCDMA). This network operated on the 2100MHz frequency band and provided greater speeds at high mobility from 144kb / s to 384kb / s and low mobility from 2Mbps. 3G increased network capacity to meet up with demand and actualized global roaming for subscribers.

3G has been a tremendous success, particularly in terms of standardization, but there are constraints and expectations that exceed it. Such issues include the high spectrum license price, high 3G network cost that makes most operators return to 2G. There are also problems with

delayed roll-out and patchy coverage. In addition, with the latest fast evolution of information systems and services, mobile devices requiring high mobility, much greater data rates and interoperability, it is necessary to harmonize all network technologies with much greater data rates at any time, (Tanko, 2014).

2.2.3 Fourth Generation Mobile Telecommunication (4G)

Soon after 3G, mobile telecommunications networks of the fourth generation are technologies constructed to meet the set of norms set by the IMT-Advanced requirements. These networks will reach elevated mobility speeds of 100Mbps and low mobility speeds of up to 1Gbps. This allows wireless systems to reach the current capabilities of wireless systems and to trigger a mobile broadband. Also the 4G network is to be an 'open wireless' system which means it should be a network with a unified core which is accessible from different wireless(access) technologies, this is aimed at harmonizing and further standardizing all the available wireless technologies. 4G networks are also 'all IP' and fully packet switched networks, (Tanko, 2014).

2.3 Google Firebase Cloud Systems

Firebase Cloud Messaging (FCM) is a cross-platform messaging solution that enables you to deliver emails reliably and free of charge. Using FCM, you can notify a client app that you can synchronize fresh email or other information. To drive user re-engagement and retention, you can send notification emails. A message can transfer a payload of up to 4 KB to a client app for use in instances such as immediate messaging, (https://www.firebase.google.com/docs/cloud-messaging).

2.3.1 How Firebase Works

There are two primary elements for sending and receiving an FCM application:

iii. A trusted environment like Firebase Cloud Functions or an App Server for building, targeting and sending emails.
iv. An iOS, Android, or web (JavaScript) client App that receives messages.

Use the Firebase Admin Software Development Kit (SDK) or the FCM server protocols to send emails. You can also use the Notifications composer to test or send marketing or commitment posts with strong integrated targeting and analytics..

Use the Firebase Messaging API and Android Studio 1.4 or greater with Griddle to write the Firebase Cloud Messaging Android client app. The guidelines on this page suppose that the measures to add Firebase to your Android project have been finished. FCM customers involve Android 4.1 or greater phones that also have the Google Play Store App installed, or an Android 4 emulator.

2.4 Java Application Programming Interface (API)

Java Application Programming Interface (API) is a list of all courses in the Java Development Kit (JDK). It involves all java packages, classes, and interfaces as well as their techniques, areas, and builders. These pre-written courses give a programmer a great deal of functionality, (http://www.wikipeadia.com).

2.5 Android Studio

Android Studio is Google's official integrated development environment for the Android operating system, built on IntelliJ IDEA software from JetBrain and specifically intended for Android

development. It can be downloaded from Windows, macros and the operating system based on Linux, (http://www.wikipeadia.com).

Android studio allows the user to apply changes and push code and resource changes the running App and in some cases, without restarting the current activity. It also has features like the intelligent code editor for writing of better codes, fast emulator to starts Apps. Android studio also comes with templates and sample Apps which includes projects and code templates that allows the addition of well established patterns such as navigator drawer and view pager.

One of the most exciting features of the Android studio is the Firebase and cloud integrations. The firebase assistant allows a developer to connect Apps to firebase and add services such as analytical, authentication and notifications with step-by-step procedures.

CHAPTER THREE

3.0 RESEARCH METHODOLOGY

3.1 Methodology

The proposed system will perform both accident crash location detection and notification. The system design block is illustrated in figure 3.1.

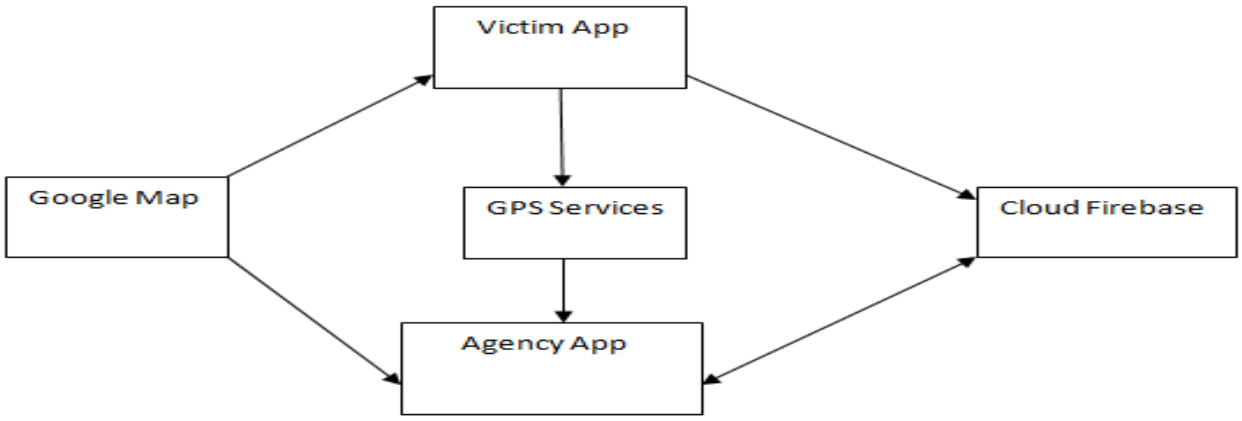


Fig. 3.1: Block design diagram.

The system is made up of two App's and a Google firebase cloud system for registering, storing and retrieval of information of the agencies and victim. The two Apps are the crash detector called user or victim's App and the agency location called agency App. Both Apps were developed using Android studio Integrated Development Environment (IDE). The victim's App uses the GPS technology and services to access the Goggle map. A Google cloud Firebase which store information on database in real-time was used to store the phone numbers and the geo-location coordinates of the agencies. The victim's smart phone with the App detects the crash, gets the geo-location coordinates through the GPS services, retrieves the agency phone number from the Firebase and notifies the closest agency to the accident site. The agency App is only used to register an agency on the Google Firebase cloud system.

3.2 System Architecture

The system architecture shown in Figure 3.2 has two parts, the client side - the mobile-phone (locating and reporting device) and the cloud system which act as a server. The smart phone on which the victim App was installed is inside the moving vehicle. As the vehicle accelerates and crashes, the victim android smart phone in conjunction with the App detects the crash if the

acceleration threshold set for the App is triggered and gets the location coordinates from GPS satellite and the link to the Google map image of the crash site from the Google map server in real-time.

The victim App then access the real-time firebase on the cloud system, fetch both the location coordinates and mobile phone number of the any of the three registered agencies that is closest to the accident location, then sends SMS containing a short message with the link of the Google map image to the nearest concerned agency phone number. The staff of the concerned agencies has full access to the features of the system such as adding or deleting an agency and accessing details of road crashes on the firebase.

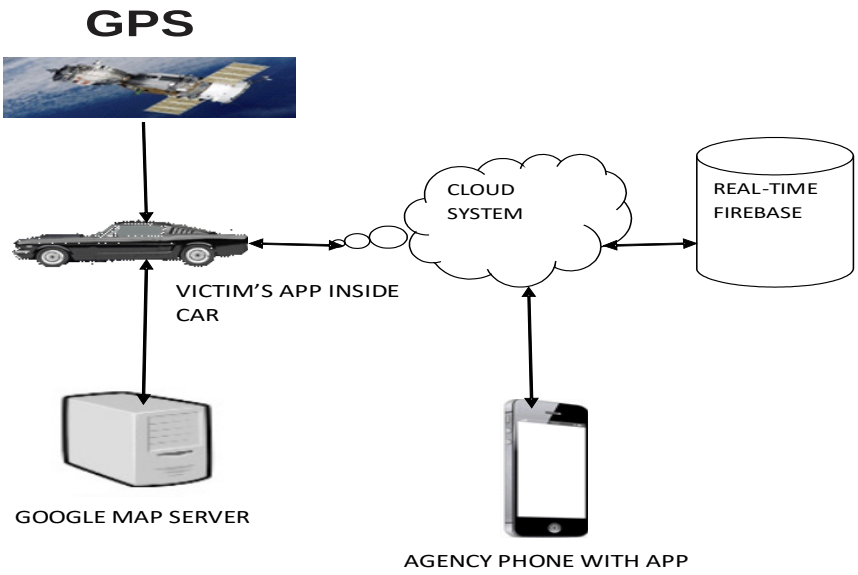


Fig. 3.2: System Architecture

3.3 System Design

The two Apps, victim and agency App were developed and implemented using Integrated Development Environment (IDE) of Android studio, with Java programming language for the Events and

XML for the interface design. The implementation was strictly on Android smart phones with minimum Application Programming Interface (API) level 16.

The system design is illustrated with the Entity-Relationship diagram in Figure 3.3; it is made up of five Entities- the Victim's App, the Agency App, the Firebase, the Concerned Agency and the User. The entities are denoted by rectangle. The entities; User, Concerned Agency and Firebase have attributes that are denoted by a circle. The User has email, phone number, name, date of birth, longitude and latitude as attributes. The Concerned Agency has agency name, email, password, phone number, longitude and latitude as attributes and the Firebase has gmail and password as attributes.

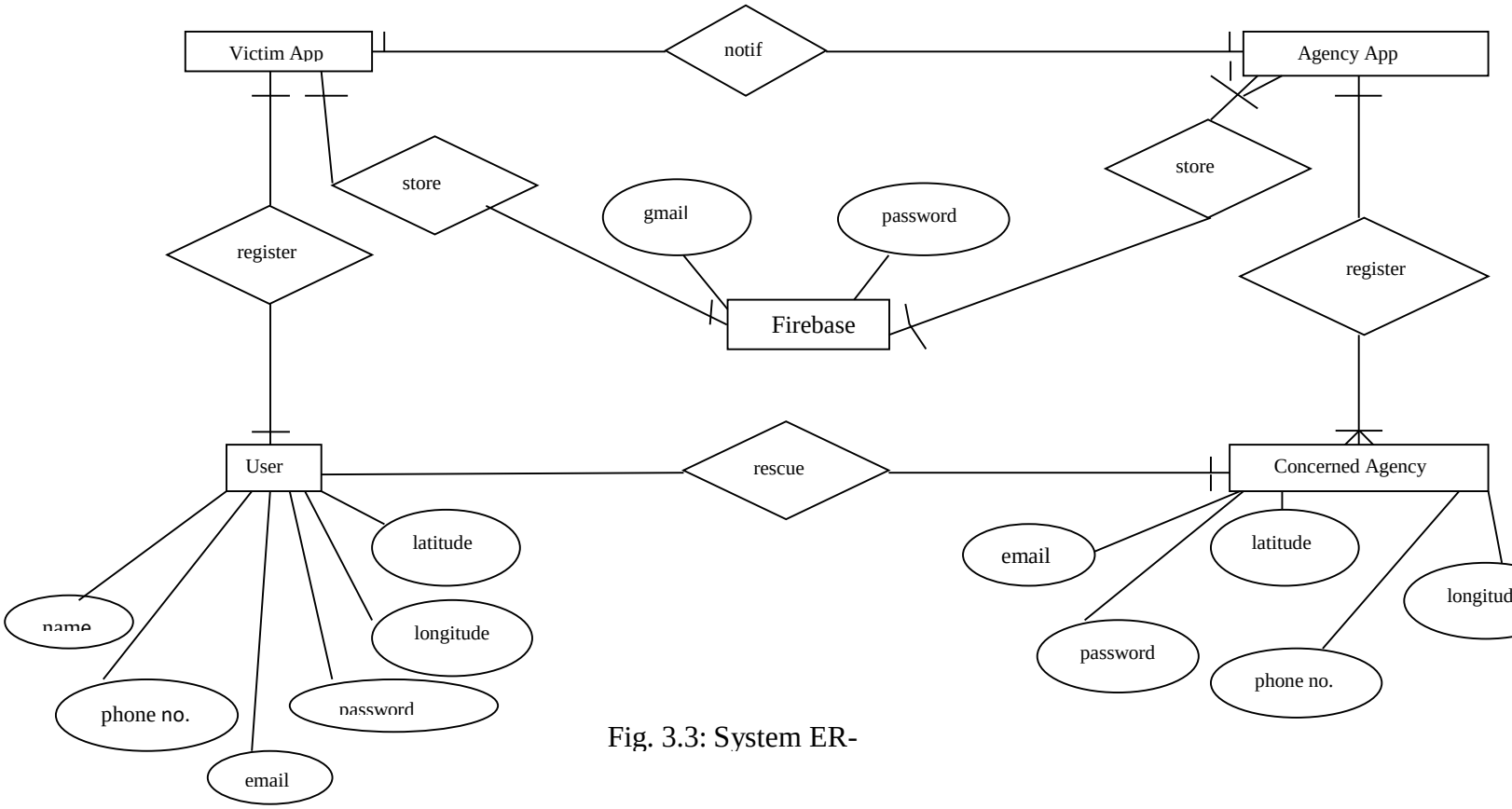


Fig. 3.3: System ER-

The relationships between the Entities are denoted by the diamond shapes. The User has register as a one-to-one Relationship with the Victim App, a relationship called store with the Firebase which is also one-to-one. The Victim App has a relationship notify with Agency App which is one-to-one too. The Agency App has a relationship with the Firebase called store which is one-to-many. The Concerned Agency has two Relationships. One with the User called rescue which is one-to-one and one with the Agency App called register which is a many-to-one Relationship.

3.3.3 The Victim App (Crash Detector)

The victim App was designed using android studio integrated developments environment. The interfaces of the App were designed with XML and java programming was used for the events. Pre-defined classes and methods were used to enable the App access GPS facility. For the victim smart phone to be able to detect a crash, an acceleration threshold was set for it through the App. This is the acceleration at which the mobile-phone detects a crash. If the acceleration experienced by the mobile-phone exceeds this threshold, then it concludes that there has been a crash. Intuitively, if the threshold is set very high, it can lead to an instance where the mobile-phone does not detect a crash even though a crash has occurred and a very low threshold might lead to false positive. Therefore, to set a threshold for the Victim's App, proper analysis is paramount.

The acceleration threshold for airbag deployment is 60Gs (around 600m/s^2), (White *et al.*, 2011). This threshold eliminates any chance of a false positive, but it ignores low speed crashes. Setting a 60G threshold will be too high for it to detect crashes in which the impact is not high enough to deploy airbags. This is because the threshold is a function of the overall vehicle design. Air bags are triggered by the accelerometer attached to the vehicle. These accelerometers are physically

installed on the car's chassis, which means that their movement directly mirrors that of the vehicle and thus experiences every force encountered by the vehicle. Smart phones, however, are likely to be held in a pocket or in a cup holder. Car safety systems are designed to reduce the force on the occupants of the car during an accident and because of this, the forces and acceleration experienced by the mobile-phone is significantly less than the forces experienced by the accelerometers in the cars, (White *et al.*, 2011).

Due to the physical environment of the mobile-phone, the threshold to be set should be significantly lower than the speed needed for airbag deployment. White *et al.* (2011) used a threshold of 4Gs, which is approximately 40m/s.^{2}To set the acceleration threshold of the victim's App, the android smart phone accelerometer designed for the App will have to be integrated with the android device to detect any form of motion. The excerpt from the code that allows this integration and makes the accelerometer of the smart phone enable in other to detect a crash is outlined in figure 3.4 from the code excerpt, the acceleration threshold set for the victim App was 15G.

CrashDetection

```
public class SensorService extends Service implements SensorEventListener
{

   // TAG to identify notification

   private st
   private Sensor accelerometer;

   private SensorManager mSensorManager;

   private double accelerationX, accelerationY, accelerationZ;

   private int threshold = 15;

      accelerationX = (Math.round(sensorEvent.values[0]*1000)/1000.0);

      accelerationY = (Math.round(sensorEvent.values[1]*1000)/1000.0);

      accelerationZ =

      if (accelerationX > threshold || accelerationY > threshold || accelerationZ
> threshold) {
```

Fig.3.4: Code excerpt for crash

3.3.4 The Agency App (Agency Locator)

The agency App was also designed with android studio like the victim's App. The App was only used to register each of the three agencies that were used to implement the system. The information supplied by the user and the agencies using the victim App and the agency App respectively were all stored on the Google cloud firebase.

3.3.3 Google Firebase Cloud System for the Victim and Agency Apps

The Firebase Cloud Messenger and the Software Development Kit were created through these steps.

i. Firebase was added to the Android project.

ii. Dependency for the Cloud Messaging Android library was added to module (app-level) Gradle file (usually app/build.gradle): implementation 'com.google.firebase:firebase-messaging:17.6.0'

iii. The victim's App manifest was edited, This is necessary if you wish to handle any messages beyond receiving background notifications on Apps such as receiving notifications in fore grounded Apps, receive data payload, send upstream messages. To access the device registration token when the application is initially started, the FCM SDK generates a registration token for the App instance. The token can be retrieved. Because the token could be rotated after initial startup, latest updated registration of the token is token taken. The registration change when:

i. The App deletes Instance ID

ii. The App is restored on a new device

iii. The user uninstalls/reinstall the App

iv. The user clears App data.

The Google firebase account was accessed through a Google mail account. The Firebase uses JSON (JavaScript Object Notation) to store data. JSON is an open-standard file format that uses human-readable text to transmit data objects consisting of attribute–value pairs and array data

types. The data are stored in a Tree-like structure. Figure 3.5 and 3.6 shows the firebase with three agencies information and victim information respectively.

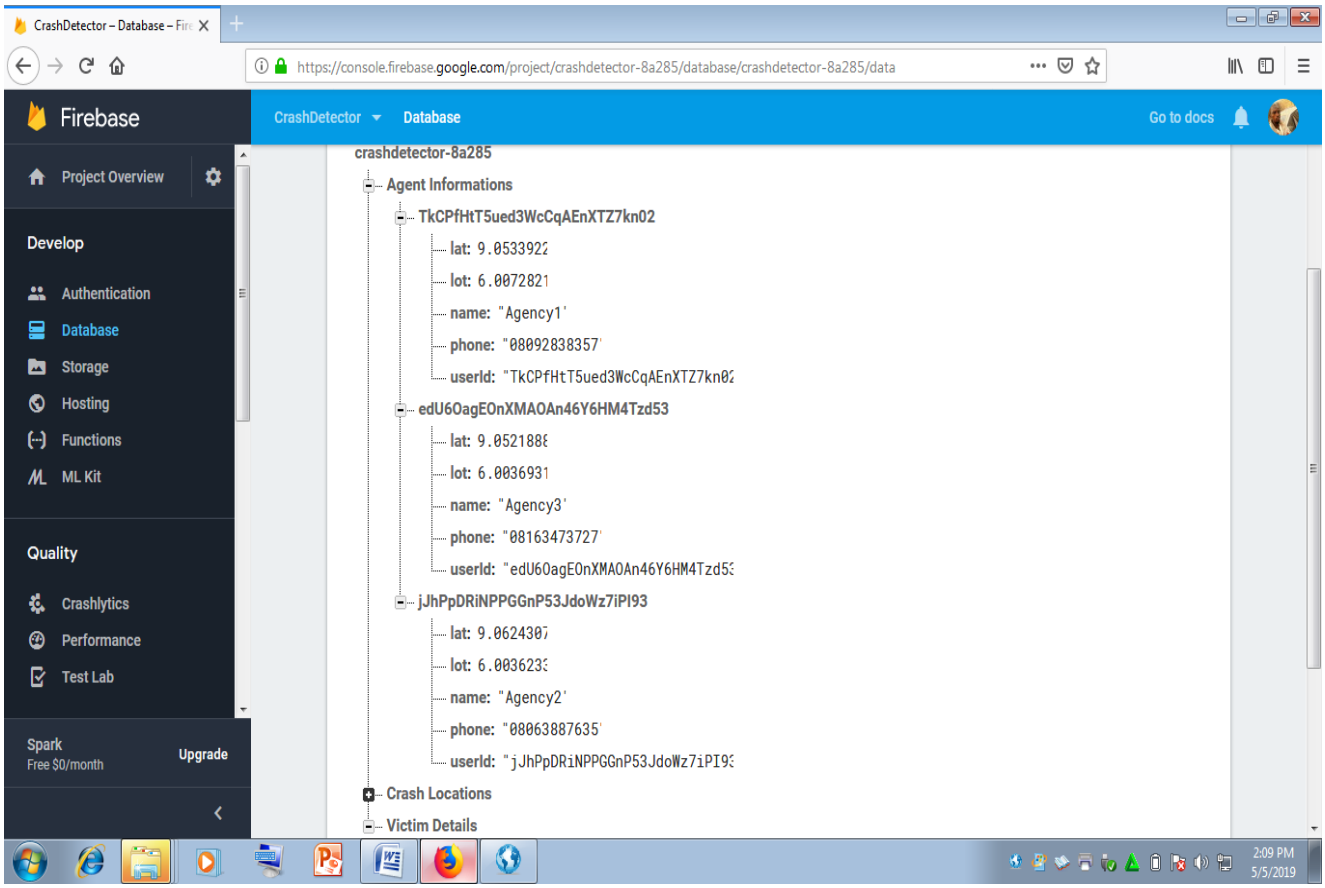


Fig. 3.5: Firebase with three agencies information

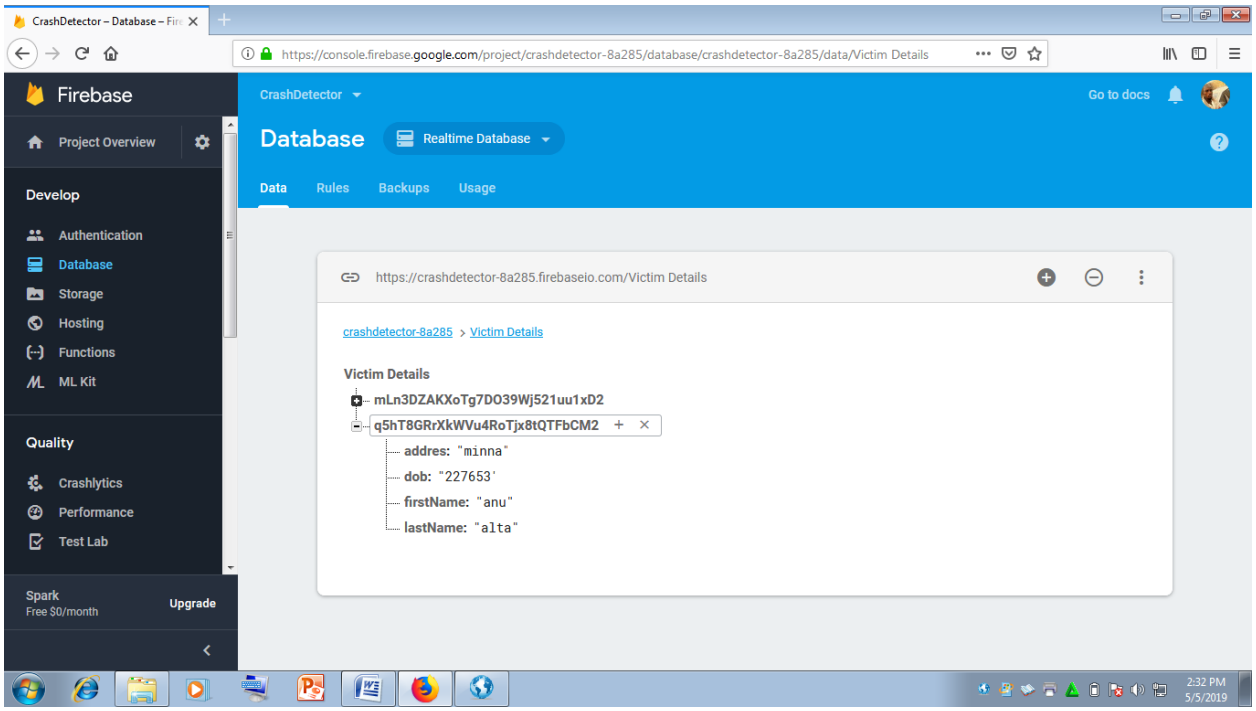


Fig. 3.6: Firebase with a victim registration details

3.3.4 Obtaining Google Map from the Google Server

The majority of android smart phones come with Google Map Apps pre-installed. The Google Maps App is activated by turning on the “Location” finding feature on the pull down menu. The Google Map relies on the smart phone’s GPS settings to know where its location is. The smart phone location actually comes with the graphical image of the location as supplied by the Google Map server.

3.3.5 Crash Location Reporting

The crash is both detected and reported by the victim’s App. When the acceleration threshold of the victim smart phone is triggered; the victim’s App retrieves the stored information of the closest concerned agency from the Google firebase, and send an SMS to it. Three concerned agencies data were stored on the Firebase. From the code excerpt shown in figure 3.7, the three agencies were

created as variables and named as agency_A, agency_B and agency_C. The mapping activity is done by the algorithm by comparing the location coordinates of the three agencies with the captured location coordinates of the crash site and send SMS to the concerned agency phone number that is closest to the crash site. If the agency registered is only one, then only the registered agency will receive the alert. The mapping comparison starts when the agencies are two or three but not more than three.

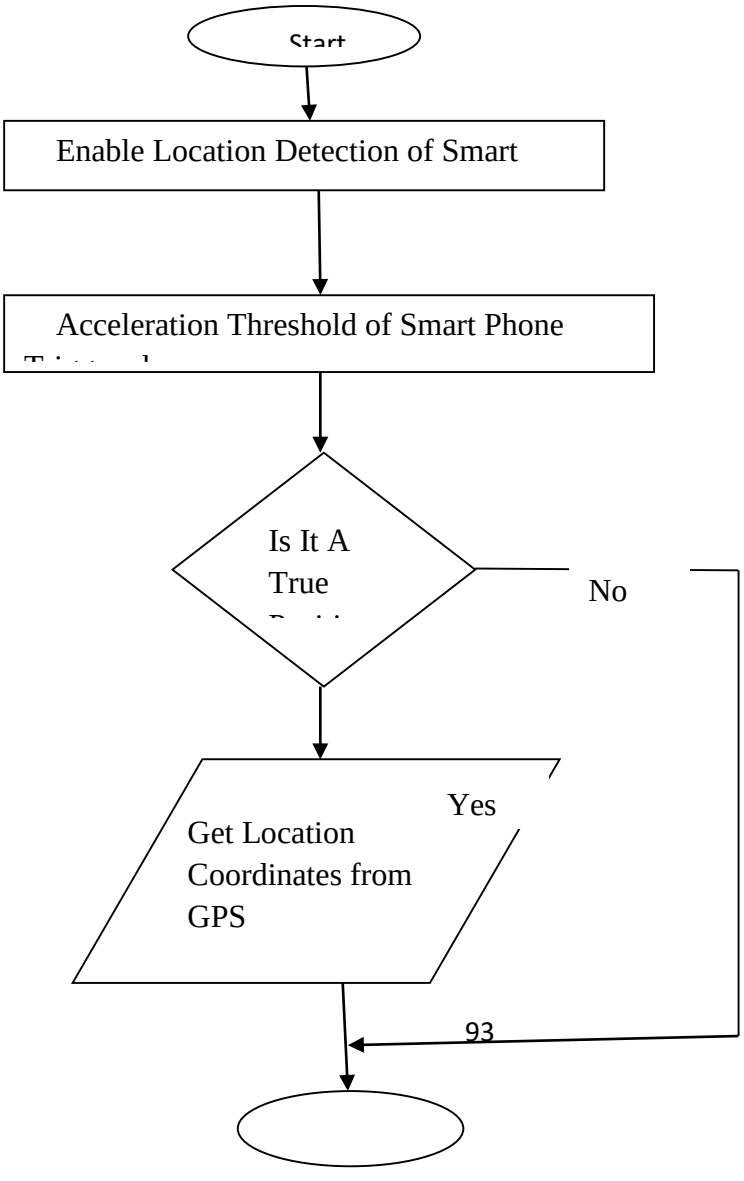
```
// CrashReporting
if(agency_A< agency_B && agent_A< agency_C){
    sendSMS(victim,ph1,crash,agency_A);
    saveToFirebase(crash);
}else if(agency_B<agency_A && agency_B<agency_C){
     sendSMS(victim,ph2,crash,agency_B);
      saveToFirebase(crash);
                    .          .
}else if(agency_C<agent_B && agency_C<agency_A){
    sendSMS(victim,ph3,crash,agency_C);
   saveToFirebase(crash);

}else if(agency_A == agency_B || agency_A==agency_C){
    sendSMS(victim,ph1,crash,agent_A);
   saveToFirebase(crash);
}else if(agency_B == agency_A || agency_B==agency_C){
```

```
            sendSMS(victim,ph2,crash,agent_B);
            saveToFirebase(crash);
        }else {
```

Fig.3.7: Code excerpt for crash reporting

3.4 System Flowcharts

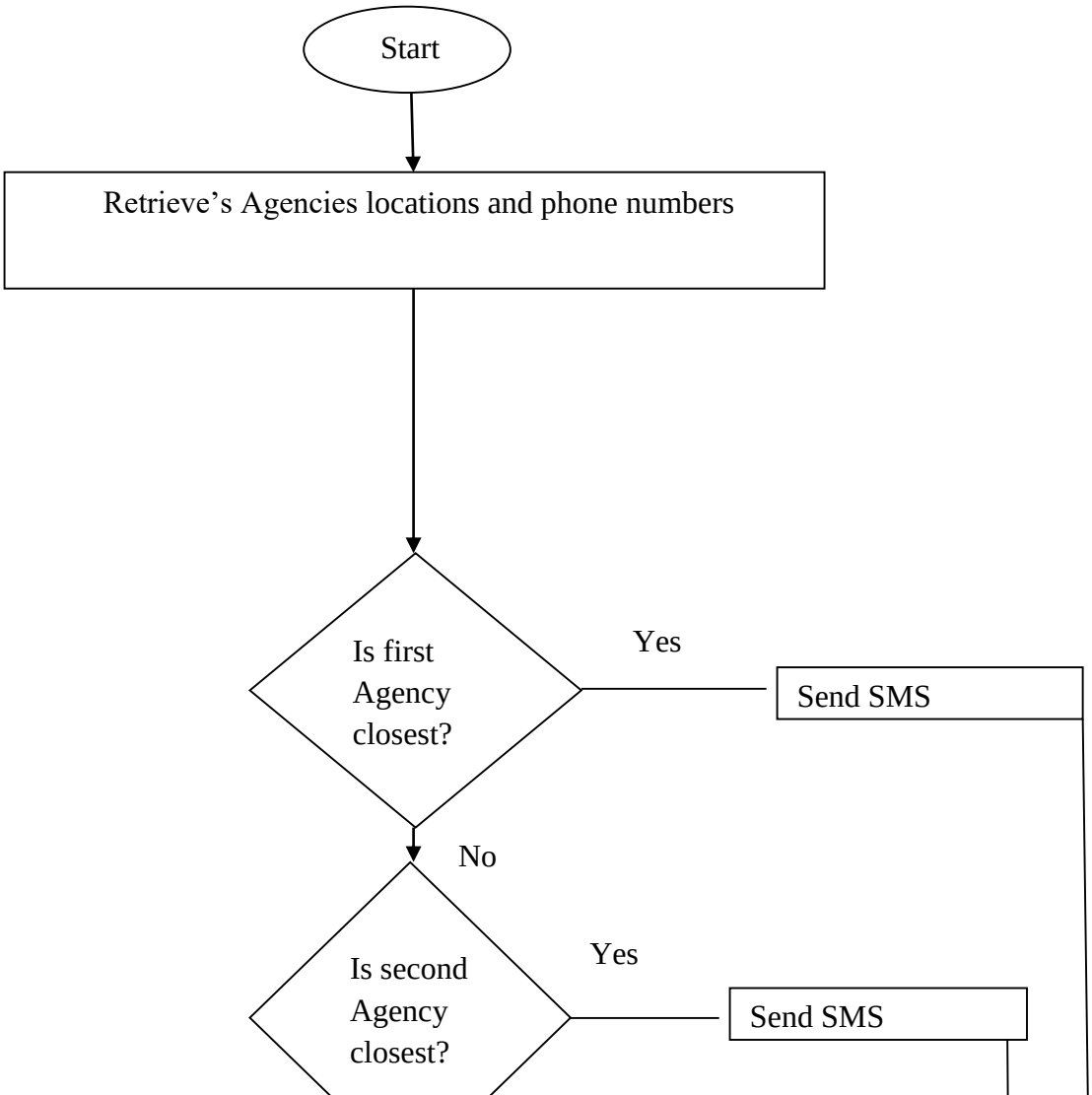
The location detection and location reporting activity of the system are represented with two system flowcharts. Figure 3.6 is the System Flowchart for the crash location detection.



Ye

Fig. 3.8: Crash location detection

After detecting the crash, the victim's App retrieves information from the firebase and sends the notification in form of SMS to the nearest agency. Figure 3.9 shows the mapping activity of the algorithm.



CHAPTER FOUR

4.0 RESULTS AND DISCUSSION

4.1 Results Obtained From the System

The results obtained from the system were analyzed. The system was tested first before the result analysis was carried out.

4.2 Testing the Proposed System

The proposed system was tested with the unit test and the system test approaches.

4.2.1Unit Test

Unit test was carried out after each of the system modules where completed. The modules are the victim App, the agency App and the Google firebase. The unit test was important to check for likely errors and the communication problem that may surface between the various system

modules. After confirming that the individual units are functioning optimally, the next test approach which is the entire system test was carried out.

4.2.2 System Test

The entire system units where tested together which is the system test. The system test required the set up for the experiment that was carried out.

4.2.2.1 Experimental Setup for the System Test

The following items were used to test the system.

vi. Four Android Smart Phones, one had the Victim App installed on it, and each of the remaining three had the Agency App installed on them.

vii. A laptop system used to access the Google Firebase account created

viii. A modem for internet connectivity for the laptop

ix. A Radio-controlled toy car.

x. Masking tape

Table 1 show the model and properties of the four android smart phones used for the experiment

Table 4.1: List of the Android Phones used and their properties

Phone Name	**Model**	**Software Version**	**Internal Storage**	**Network Type**			**App installed**	
				2G	**3G**	**4G**	**Victim's App(3.14MB)**	**Agency App(2.72 MB)**
Nokia 5		7.1	16G	✓	✓	✓	✓	

INFINIX X559C	7.0	32G	✓	✓	✓	✓		✓
Techno LA6	7.0	16G	✓	✓				✓
Techno K7	7.0	16G	✓	✓				

A long corridor was used as the traveling path for the radio-controlled vehicle. The path was divided into three sections; each position was 15 meters or more away from the next section. The three smart phones INFINIX 559C, Techno LA6 and Techo K7 were stationed one each on the three positions.

From table 4.1, the three smart phones stationed on the travelling path were the ones with the agency App installed on them. Each of the smart phones on the path represented the location of an emergency center. The agencies registrations were carried out from the various locations the smart phones were stationed in other to be able to obtain there static geo-location coordinates addresses. The victim's registration was done at a random location along the travelling path using the Nokia 5 smart phone.

4.2.2.3 Registration

The victim and agency Apps were used to register a victim and the agencies information's respectively that were stored on the Google firebase. The registration was simple. Figure 4.1 and 4.2 shows the display windows of the both victim and the agency App respectively after successfully installing them.

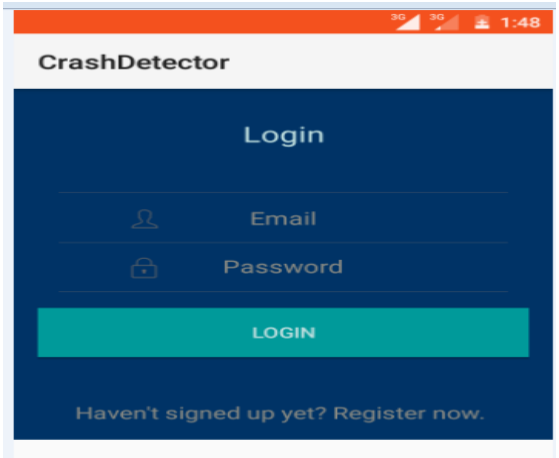
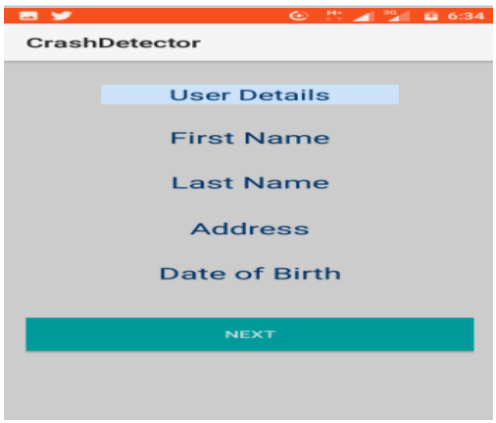


Fig. 4.1: Victim's App display window

Fig. 4.2: Agency App display window

A new user was registered by clicking Register now. After clicking register now, the registration window came up where all the user details that was needed was supplied by the user. For the agency App, the register button was clicked and the agencies information were registered. Figure 4.3 and 4.4 shows the registration windows for the victim and agency Apps respectively.



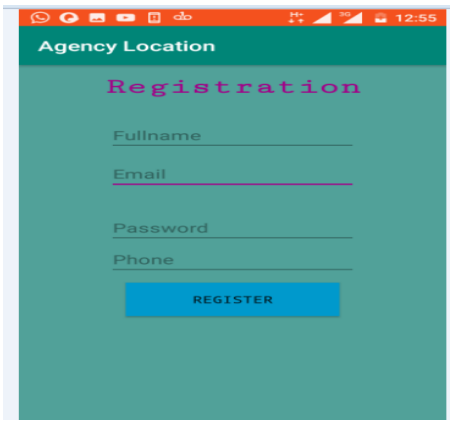


Fig. 4.3: Victim registration window

Fig. 4.4: Agency registration window

Tables 4.2 and 4.3 shows the input specification required for both the victim and the agency App registration respectively.

Table 4.2 Input specifications for the registration window of victim App

User Detail	Data Type
First Name	String
Last Name	String
Address	Alphanumeric
Date Of Birth	Numeric And Special Characters

Table 4.3: Agency App input specification for registration

Agency (Agent) Info	Data Type
Name Of Agency	String
Email	Alphanumeric And Special Characters
Password	Numeric
Phone Number	Numeric

After the registration, the user logged in with a Gmail account, phone number and a password of eight numeric digits. Gmail and numeric password were used to log onto the agency App.

4.2.2.3 Vehicle Crash Setup for the Experiment

The Radio-controlled toy car that was used to simulate the crash has a speed of around 30km/h, Figure 4.3 shows the different views of the vehicle and how the smart phone with the victim App was strapped on the vehicle by a masking tape. The smart phone cellular network type was selected and the data service was switched on; the find location feature was also activated.

Before the vehicle was accelerated for the crash, the victim App was opened on the smart phone and a button was activated to start tracking the location of the smart phone as it accelerates by the

App. The vehicle was accelerated for about 4seconds to attain top speed before it was crashed on a padded surface. The crash was strong enough to trigger the acceleration threshold set.

Fig. 4.5: Radio-controlled car set up for experiment

As soon as the acceleration threshold was triggered, a window shows up with an alarm and a 30seconds timer. The purpose of having the alarm and the timer is to be able to detect fake crash or false positive, a situation where the system detect a crash where there was no crash. The timer counts down from 30seconds, if it is not stopped, after the countdown; the victim's App accessed the firebase and fetches the geo-locations coordinates of the three registered agencies and compare their locations with the location of the crashed vehicle. It then mapped the crash location address to the address of the closest agency to the crash and sends SMS to that agency phone number in real-time. Figure 4.6 and 4.7 shows the window of the countdown timer and the short SMS sent to the closest agency respectively. The SMS contained the link to the Google map server where image of the location of crash site was found.

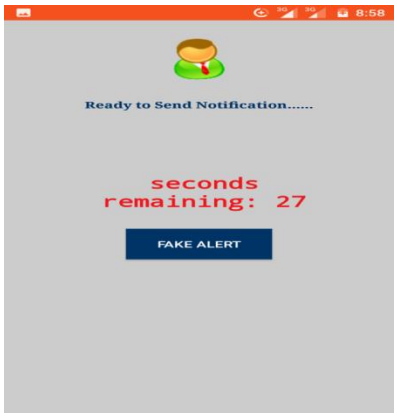


Fig. 4.6: Countdown timer of the

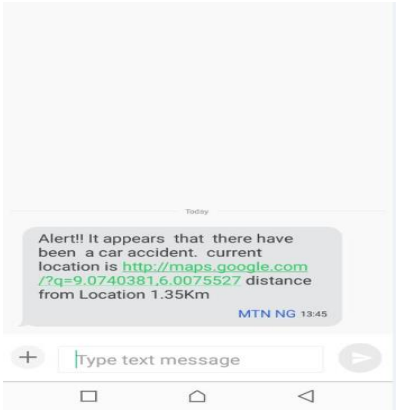


Fig. 4.7: Crash details

The link to the location was clicked and the imagery of the crash location was displayed as shown in Figure 4.6.

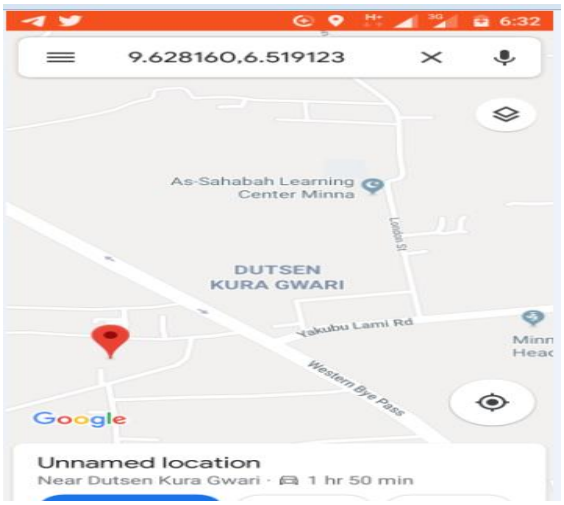


Fig. 4.8: Google map-view of crash location

4.3 Discussion of the Obtained Results

To discuss the result, the system performance was evaluated using two performance indicators. The first indicator used was the confusion matrix. The confusion matrix was used to store the result

of the classifier from 100 crash scenarios. The system detected the class of each instance; this classification is shown in Table 4.4

Table 4.4: The system confused matrix

TOTAL CRASH	**PREDICTED ROAD CRASH**		**TOTAL**
ACTUAL ROAD CRASH	TRUE POSITIVE= 55	FALSE NEGATIVE = 4	59
	FALSE POSITIVE =5	TRUE NEGATIVE = 36	41
TOTAL	60	40	100

The following performance metrics were calculated from table 4.4

i. $$\text{Accuracy} = \frac{TN + TP}{TP + TN + FP + FN} = 91\%$$

ii. $$\text{Sensitivity} = \frac{TP}{TP + FN} = 93\%$$

iii. $$\text{Precision} = \frac{TP}{TP + FP} = 92\%$$

iv. $$\text{Specificity} = \frac{TN}{TN + FP} = 87\%$$

The second performance indicator that was used was the cellular mobile network type used by the smart phone with the Victim App during the System Test. The various network type used by mobile devices are 2G, 3G and 4G networks. From Table 4.1, all the four android smart phones have 2G and 3G network. Only Nokia 5(Victim App) and the INFINIX X559C had 4G networks.

Due to the unavailability of the 4G networks in most android smart phones and the smaller range it covers in comparison to both 2G and 3G network, the 2G and 3G networks were used to obtain real-time delivery of the SMS to the phones with the agency Apps. Variations where observed in the delivery time and was recorded.

Tables 4.5 and 4.6 shows the results obtained from 5 crashes each with 2G and 3G network respectively. Five crashes where simulated with the 2G network and another five with 3G network enabled on the Nokia 5 phone respectively, variations were observed in delivery time of the SMS received .

Table 4.5: Crash result showing SMS delivery time with 2G Networks.

Crash No.	Delivery Time in Seconds
1	8.7
2	7.2

3	7.4
4	5.6
5	5.8

Figure 4.7 shows the effect of the time variation recorded on a graph for the 2G network. The graph pattern is not as smooth as the values obtained and showed on figure 4.8 which is for 3G network. It took a longer time to deliver the SMS with the 2G network in comparison with the 3G network. This as a result, shows why the graph in figure 4.8 was smoother. The vertical axis represent the delivery time in seconds while the horizontal axis showed the number of crashes recorded.

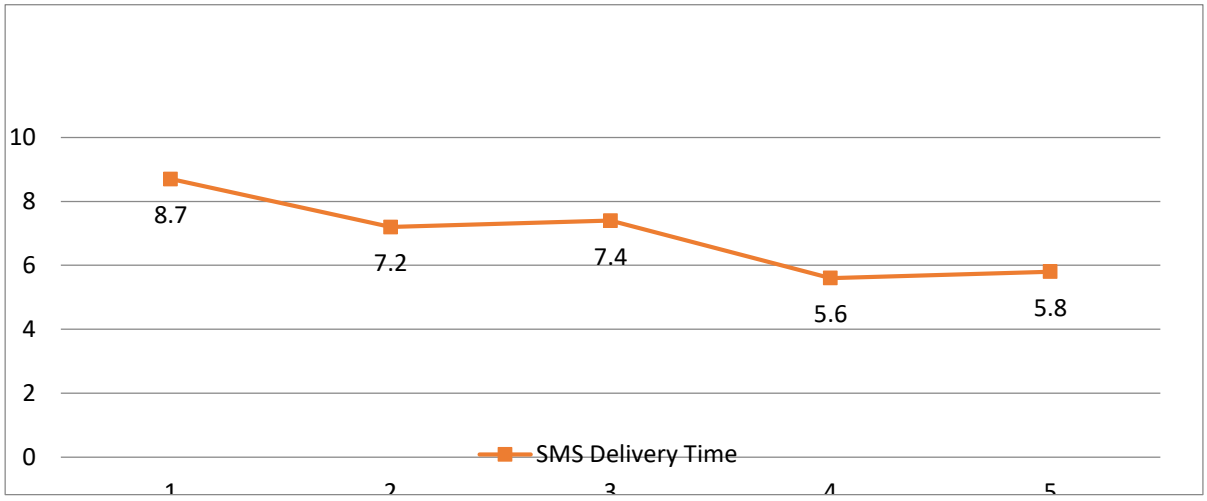


Fig. 4.9: SMS real-time delivery for 2G

Table 4.6: Crash result showing SMS delivery time with 3G Networks.

Crash No.	Delivery Time in Seconds

1	4.6
2	4.2
3	3.9
4	3.5
5	4.2

Figure 4.8 in the other hand represent a smoother steep, the delivery time by the 3G network was better than that of the 2G network.

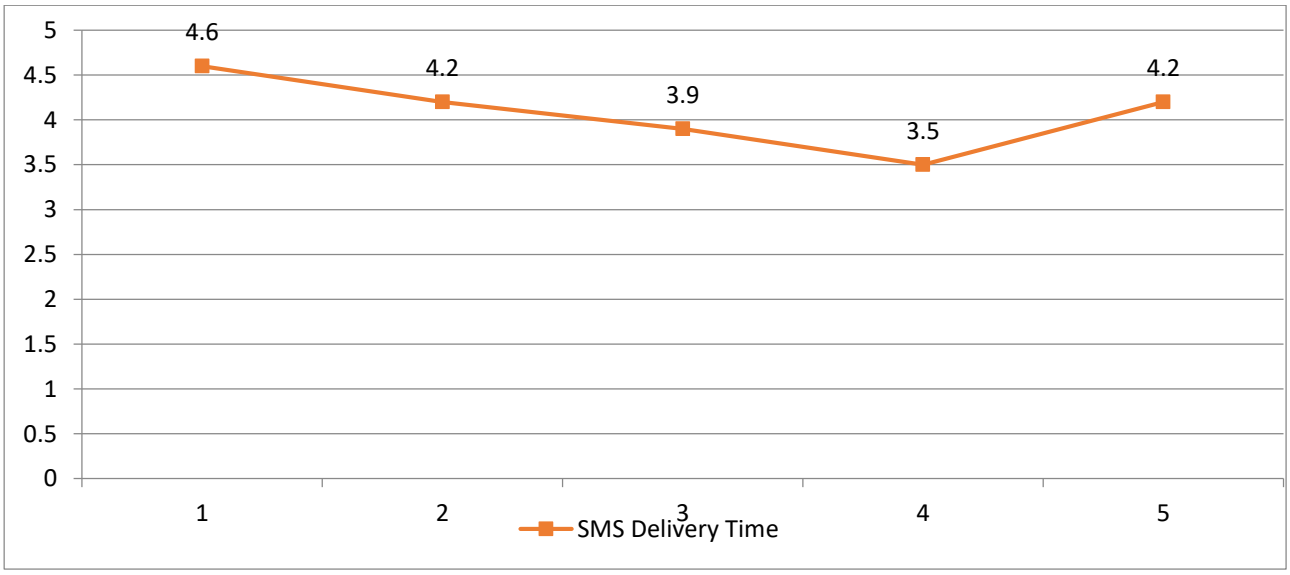


Fig. 4.10: SMS real-time delivery for 3G

Standard SMS rate was charged by the network service provider from the mobile phone number inside the victim's smart phone.

CHAPTER FIVE

5.0 CONCLUSION AND RECOMMENDATION

5.1 Conclusion

A mobile phone-based vehicle crash location detection and alerting system using android smart phone and GPS technologies was developed. The system was successfully implemented as both the Apps were successfully installed on different Android phones and each successfully retrieved the devices location using the devices inbuilt GPS triggered by its accelerometer (i.e., devices change in acceleration) and SMS was sent to the agency closest to the crash site successfully. This was achieved by the algorithm developed and integrated in the Victim's App for the system. The evaluation of the system was carried out and results of experiments were presented using a

confusion matrix and the cellular network type. The confusion matrix results obtained showed high Sensitivity, adequate accuracy and precision.

5.2 Recommendation

Two recommendations were made at the end of the research work which was:

i. Since the developed system used static mode of addressing for the registration of the three agencies used. The location where an agency was stationed during its registration is that same locations coordinates the algorithm fetched and perform the mapping operation. Using dynamic addressing will make the system more flexible where an already registered agency can change location without re-registering again. It is recommended that dynamic addressing should be used because that will make the agency to move around like ambulances and Red Cross vehicles do.

ii. Secondly, the network type signifies how quick the victim App retrieved the required information from the Google Firebase that was sent in the SMS. Due to the unavailability of 4G network, only 2G and 3G were used for the system. Since 4G is a faster network, it is recommended that it should be used.

5.3 Contribution to knowledge

The system designed was not only able to detect crash, it was able to send SMS to a first responder that is closest to the crash site. This will make it possible for the victim's that needs urgent help get prompt attention. The ability of the designed system to locate the closest emergency center from the crash site was the major contribution to knowledge.

5.4 Suggestion for Future Work

For future work, the addresses of the emergency centers can be made dynamic, this will allow the emergency centers that are mobile i.e. ambulances, to still receive the crash alert even when there

are moving around and 4G or 5G cellular network type should be used for the system in other to reduce the SMS delivery time of the system.

REFERENCES

Aloul, F., Zualkernan, I., Abu-Salma, R., Al-Ali, H. & Al-Merri, M. (2014). iBump: Smartphone application to detect car accidents, *IEEE International Conference on Industrial Automation, Information and Communications Technology (IAICT)*. 12(2), 52-56.

Azeez, R.A., Ogunrinde, M. A., & Sakirullah, O. A. (2015). A Web-based accident reporting and tracking system using sensor technology. *International Journal of Advances in Engineering and Technology*. 8(5), 678-684

Baramy, N., Singh, K. T., Jadhav, A., Javir, A. & Tarleka, S. (2016). Accident detection & alerting system, *International Journal of Technical Research and Applications.* 39, 8-11.

Dang, T.T., Truong, H., Tran, K.D (2016). Automatic fall detection using smart phone acceleration sensor. *(IJACSA) International Journal of Advanced Computer Science and Applications.*7(12), 12-17

Documentation for Firebase Cloud System. Retrieved from https://firebase.google.com/docs/cloud-messaging-2019-6

GB Blog Official, (2018). Retrieved from https://www.gearbest.com/blog/how-to/gps-and-gprs-tracking-2019-8

Goh, K.N., Jaafar, J., Mustapha, E.E.,Goh, E.T (2014). Automatic accident location detection System. *4th World Congress on Information and communication.* 2(8), 7-9

DOI:10.1109/WICT.2014.7077303

I.O.P (2014). How GPS works. Retrieved from http://www.physics.org/article-questions.asp?id=2019-10

Kaladevi, P., Kokila, T., Narmatha, S., & Janan, V. (2014). Accident Detection Using Android Smart Phone. *International Journal of Innovative Research in Computer And Communications Engineering*. 2(1), 2367-2372.

Khalil, U., Nasir, A., Javid, T., Raza, S. A., & Siddiqui, A. (2018). Automatic road accident detection using ultrasonic sensor. *International Symposium on Wireless Systems and Networks*. 5, 1-6, DOI: 10.1109/INMIC.2018.8595541

Narendar, D.S & Teja, R. (2013).Vehicle speed limit alerting and crash detection system at various zones. *International Journal of Latest Trends in Engineering and Technology*. 2(1), 108-113.

Njuguna, P. N. (2012). Instant GPS based motor vehicle accident detection and reporting. International Journal of I.C.T. *University of Nairobi School of Computing and Informatics*, 4, 12-15

Tanko, A. L. (2014). Evolution of mobile telecommunication with respect to 1G 2G, 3G and 4G. Retrieved from https://www.ncc.gov.ng/thecommunicator/2019-6

White, J., Thompson, C., Turner, H., Dougherty, B., & Schmidt, D.C. (2011). Wreckwatch: Automatic traffic accident detection and notification with smart phones. *Mobile Networks and Applications*, 16(3), 285-303.

WHO, "World Report on Road Traffic Injury Prevention: Summary" (2012). *World Health Organization*, Geneva, Switzerland.

Yogesh , J. I. & Rane, U.A. (2014). A Review on ARM7 based accident detection using GSM, GPS and MEMS. International Journal of Engineering Sciences & Research Technology. *Department of Electronics and Telecommunication Engineering , Shegaon, Maharashtr SSGMCE*

Zahradnik, F. (2017). An overview on how GPS technology works. Retrieved from https://www.lifewire.com/more-2018-7

APPENDIX A (SOURCE CODES)

```
package com.example.codesigood.crashdetector;

import android.support.annotation.NonNull;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.support.v4.view.GravityCompat;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;

public class DashboardActivity extends AppCompatActivity  {

    private final int MY_PERMISSION_REQUEST_CODE = 1;
    private PermissionHandler mPermissionHandler;
```

```
    private FirebaseAuth firebaseAuth;
    private FirebaseUser firebaseUser;
    private ServiceHandler mServiceHandler;
    private static boolean isTracking;
    private Button buttonToggleTracking;
    private DBEmergency mDatabase;
    private LocationManager locationManager;
    private DrawerLayout mDrawerLayout;
    private ActionBarDrawerToggle mDrawerToggle;
    private CharSequence mDrawerTitle;
    private CharSequence mTitle;
    ExpandableListAdapter listAdapter;
    ExpandableListView mDrawerexpList;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_dashboard);

        locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
       // custom_font = Typeface.createFromAsset(getAssets(), "AvenirNextLTPro-
MediumCn.otf");
        mPermissionHandler = new PermissionHandler(this);
        firebaseAuth = FirebaseAuth.getInstance();
        mServiceHandler = new ServiceHandler(this);
        isTracking = false;
        buttonToggleTracking = (Button) findViewById(R.id.buttonToggleTracking);
        CustomToastActivity.CustomToastActivity(this);
        mDatabase = new DBEmergency(this);

        setupFirebase();

        prepareListDataSignin();
        listAdapter=new
ExpandableListAdapter1(this,listDataHeader,login_icons,custom_font,custom_font);

        mTitle = mDrawerTitle = getTitle();
        mDrawerexpList = (ExpandableListView)findViewById(R.id.left_drawer);
        mDrawerexpList.setGroupIndicator(null);
      //  custom_font = Typeface.createFromAsset(getAssets(), "AvenirNextLTPro-
MediumCn.otf");
        mDrawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);

        // set a custom shadow that overlays the main content when the drawer opens
        mDrawerLayout.setDrawerShadow(R.drawable.drawer_shadow, GravityCompat.START);
```

```
 //set drawer expandable list adapter
      mDrawerexpList.setAdapter(listAdapter);
 //Creation of expandable listView
      mDrawerexpList.setOnGroupClickListener(new
ExpandableListView.OnGroupClickListener() {

         @Override
         public boolean onGroupClick(ExpandableListView parent, View v,
                              int groupPosition, long id) {
            if (groupPosition == 0) {
               // Already in this Activity
               mDrawerLayout.closeDrawer(mDrawerexpList);
            }
            if (groupPosition == 1) {
               finish();
               Intent intent=new Intent(DashboardActivity.this, MyEmerContActivity.class);
               startActivity(intent);
               mDrawerLayout.closeDrawer(mDrawerexpList);
            }
            if (groupPosition == 2) {
               finish();
               Intent intent=new Intent(DashboardActivity.this, MyAccount.class);
               startActivity(intent);
               mDrawerLayout.closeDrawer(mDrawerexpList);
            }
            if (groupPosition == 3){
               logout();
            }
            return false;
         }
      });
      // Listview Group expanded listener
      mDrawerexpList.setOnGroupExpandListener(new
ExpandableListView.OnGroupExpandListener() {

         @Override
         public void onGroupExpand(int groupPosition) {
         }
      });

      // Listview Group collasped listener
      mDrawerexpList.setOnGroupCollapseListener(new
ExpandableListView.OnGroupCollapseListener() {

         @Override
         public void onGroupCollapse(int groupPosition) {
```

```
        }
    });

    final android.support.v7.widget.Toolbar toolbar = (android.support.v7.widget.Toolbar)
findViewById(R.id.toolbar);
    mDrawerToggle = new ActionBarDrawerToggle(

  private void setupFirebase() {
    firebaseAuth = FirebaseAuth.getInstance();
    firebaseUser = firebaseAuth.getCurrentUser();
    if (firebaseUser == null) {
      finish();
      startActivity(new Intent(this, LoginScreenActivity.class));
    }
  }

  public void logout()
  {
    try {
      firebaseAuth.signOut();
      finish();
      startActivity(new Intent(this, LoginScreenActivity.class));
    } catch (Exception e) {
      Toast.makeText(this, "Unsuccessful", Toast.LENGTH_SHORT).show();
      e.printStackTrace();
    }
  }

  @Override
  public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
@NonNull int[] grantResults) {
    switch (requestCode) {
      case MY_PERMISSION_REQUEST_CODE:
        if (mPermissionHandler.handleRequestResult(requestCode, permissions, grantResults))
{
          toggleTracking();
        }
        break;
      default:
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    }
  }

  private boolean locationServicesStatusCheck() {
```

```
        final LocationManager locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);

        if (locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER)) return true;

        AlertDialog.Builder builder = new AlertDialog.Builder(DashboardActivity.this);
        builder.setTitle("Enable GPS")
                .setMessage("This function needs your GPS, do you want to enable it now?")
                .setIcon(android.R.drawable.ic_menu_mylocation)
                .setCancelable(false)
                .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface, int i) {
                        startActivity(new
Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS));
                    }
                });
        AlertDialog dialog = builder.create();
        dialog.show();


        return false;
    }

    private boolean hasContact() {
        String email = firebaseUser.getEmail();
        List<EmerContact> contact = mDatabase.getContact(email);
        if (contact.isEmpty()) {
            CustomToastActivity.showCustomToast("Please add at least 1 Emergency Contact");
            return false;
        } else {
            return true;
        }
    }

    private void prepareListDataSignin() {
        listDataHeader =new ArrayList<String>();
        //listDataChild = new HashMap<String, List<String>>();

        // Adding group data
        listDataHeader.add("Dashboard");
       // listDataHeader.add("Emergency Contacts");
        listDataHeader.add("My Account");
        listDataHeader.add("Log Out");

    }
```

//SendSMS Activity

```
package com.example.codesigood.crashdetector;

import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.support.v4.view.GravityCompat;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support.v7.app.AlertDialog;
import                                          android.support.v7.app.AppCompatActivity;import
android.widget.ExpandableListAdapter;
import android.widget.ExpandableListView;
import android.widget.TextView;
import android.widget.Toast;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

public class DashboardActivity extends AppCompatActivity  {

    private final int MY_PERMISSION_REQUEST_CODE = 1;
    private PermissionHandler mPermissionHandler;

    private FirebaseAuth firebaseAuth;
    private FirebaseUser firebaseUser;
    private ServiceHandler mServiceHandler;
    private static boolean isTracking;
    private Button buttonToggleTracking;
    private DBEmergency mDatabase;
    private LocationManager locationManager;
    private DrawerLayout mDrawerLayout;
    mPermissionHandler = new PermissionHandler(this);
        firebaseAuth = FirebaseAuth.getInstance();
        mServiceHandler = new ServiceHandler(this);
        isTracking = false;
        buttonToggleTracking = (Button) findViewById(R.id.buttonToggleTracking);
        CustomToastActivity.CustomToastActivity(this);
        mDatabase = new DBEmergency(this);
  setupFirebase();
prepareListDataSignin();
        listAdapter=new
ExpandableListAdapter1(this,listDataHeader,login_icons,custom_font,custom_font);
mTitle = mDrawerTitle = getTitle();
```

```
        mDrawerexpList = (ExpandableListView)findViewById(R.id.left_drawer);
        mDrawerexpList.setGroupIndicator(null);
      //          custom_font          =          Typeface.createFromAsset(getAssets(),          "AvenirNextLTPro-
MediumCn.otf");
        mDrawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);

        // set a custom shadow that overlays the main content when the drawer opens
        mDrawerLayout.setDrawerShadow(R.drawable.drawer_shadow, GravityCompat.START);

    private void setupFirebase() {
        firebaseAuth = FirebaseAuth.getInstance();
        firebaseUser = firebaseAuth.getCurrentUser();
        if (firebaseUser == null) {
            finish();
            startActivity(new Intent(this, LoginScreenActivity.class));
        }
    }

    public void logout()
    {
        try {
            firebaseAuth.signOut();
            finish();
            startActivity(new Intent(this, LoginScreenActivity.class));
        } catch (Exception e) {
            Toast.makeText(this, "Unsuccessful", Toast.LENGTH_SHORT).show();
            e.printStackTrace();
        }
    }


Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS));
                    }
                });
        AlertDialog dialog = builder.create();
        dialog.show();
        return false;
    }

    private boolean hasContact() {
        String email = firebaseUser.getEmail();
        List<EmerContact> contact = mDatabase.getContact(email);
        if (contact.isEmpty()) {
            CustomToastActivity.showCustomToast("Please add at least 1 Emergency Contact");
            return false;
        } else {
```

```
            return true;
        }
    }

    private void prepareListDataSignin() {
        listDataHeader =new ArrayList<String>();
        //listDataChild = new HashMap<String, List<String>>();

        // Adding group data
        listDataHeader.add("Dashboard");
       // listDataHeader.add("Emergency Contacts");
        listDataHeader.add("My Account");
        listDataHeader.add("Log Out");

    }


    public boolean onCreateOptionsMenu(Menu menu) {
        return super.onCreateOptionsMenu(menu);
    }
```

// SensorActivity

```
package com.example.codesigood.crashdetector;
public class SensorService extends Service implements SensorEventListener

    // TAG to identify notification
    private static final int NOTIFICATION = 007;
 // IBinder object to allow Activity to connect
    private final IBinder mBinder = new LocalBinder();

    // Sensor Objects
    private Sensor accelerometer;
    private SensorManager mSensorManager;

    private double accelerationX, accelerationY, accelerationZ;

    private int threshold = 15;

    // Notification Manager
    private NotificationManager mNotificationManager;

    @Override
    public IBinder onBind(Intent intent) {
        // TODO: Return the communication channel to the service.
```

```
        return mBinder;
    }

    @Override
    public boolean onUnbind(Intent intent) {

        return super.onUnbind(intent);
    }

    public class LocalBinder extends Binder {
        public SensorService getService() {
            return SensorService.this;
        }
    }

    public void onSensorChanged(SensorEvent sensorEvent) {
        accelerationX = (Math.round(sensorEvent.values[0]*1000)/1000.0);
        accelerationY = (Math.round(sensorEvent.values[1]*1000)/1000.0);
        accelerationZ = (Math.round(sensorEvent.values[2]*1000)/1000.0);
        /*** Detect Accident ***/
        if (accelerationX > threshold || accelerationY > threshold || accelerationZ > threshold) {
            Intent mIntent = new Intent();
            mIntent.setClass(this, SendSMSActivity.class);
            mIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            mSensorManager.unregisterListener(this);                    // Unregister sensor when not
in use
            mNotificationManager.cancel(NOTIFICATION);
            stopSelf();
            startActivity(mIntent);
```

AGENCY

//Agent

```
package com.example.codesigood.agency;

public class Agents {

    private String name,phone,userId;
    private double lat,lot;

public Agents(String name, String phone,String userId,double lat,double lot)
    {
        this.name = name;
        this.phone = phone;
        this.userId = userId;
```

```
        this.lat = lat;
        this.lot =lot;
    }
    public String getName() {
        return name;
    }

    public String getPhone(){return phone;}
    public String getUserId(){return userId;}
    public double getLat(){return lat;}

    public double getLot() {
        return lot;
    }
}
```

// Register

```
package com.example.codesigood.agency;
import org.w3c.dom.Text;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.TimeZone;

public class Register extends AppCompatActivity {
    private final int LOCATION_REQUEST_CODE = 199;
    EditText fullname,email,password,phone;
    private ProgressDialog progressDialog;
    private FirebaseDatabase fireDatabase;
    private FirebaseAuth fireAuth;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);
        initial();
        isTracking = false;
        mPermissionHandler = new PermissionHandler(this);
        startTrackingWrapper();
        locationUpdater();
    }
    public void initial(){
```

```
        phone = (EditText) findViewById(R.id.phone);
        fullname = (EditText)findViewById(R.id.regName);
        email = (EditText) findViewById(R.id.regEmail);
        password = findViewById(R.id.regPassword);
        progressDialog = new ProgressDialog(this);
        fireAuth =FirebaseAuth.getInstance();
    }

   public void register(View view){
       final String name = fullname.getText().toString().trim();
        final String emails = email.getText().toString().trim();
       final String pass = password.getText().toString().trim();
       final String phones =phone.getText().toString().trim();
        if(!Patterns.EMAIL_ADDRESS.matcher(emails).matches())
        {
           Toast.makeText(getApplicationContext(),"Sorry!                                    invalid
email",Toast.LENGTH_LONG).show();
           return;
        }
        if(TextUtils.isEmpty(emails)){
              return;
        }
        if(TextUtils.isEmpty(name))
        {
           return;
        }
        if(TextUtils.isEmpty(pass))
        {
           return;
        }

           progressDialog.setMessage("Registering....  Agent!");
           progressDialog.show();
locationUpdater();
       fireAuth.createUserWithEmailAndPassword(emails,pass)
                .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
              @Override
              public void onComplete(@NonNull Task<AuthResult> task) {
                 if(task.isSuccessful())
                 {
                    fireAuth.signInWithEmailAndPassword(emails,pass);
                    FirebaseUser user  = fireAuth.getCurrentUser();
                    dataReference          =          FirebaseDatabase.getInstance().getReference("Agent
Informations");
                    String key = user.getUid();
```

```
                Agents agents = new Agents(name,phones,key,mCurrentLocation.getLatitude(),mCurrentLocation.getLongitude());
                dataReference.child(key).setValue(agents);
                // saveToFirebase();
              progressDialog.dismiss();
                Toast.makeText(getApplicationContext(),"Registration Successfull !",Toast.LENGTH_LONG).show();
                startActivity(new Intent(getApplicationContext(), MainActivity.class));
                finish();
                //finish();

              }else{

                Toast.makeText(getApplicationContext(),"Faild to register Agent!",Toast.LENGTH_LONG).show();
              }

            }
        });

  }
   public void locationUpdater()
   {
      locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
      if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
         return;
      }
      if (locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER)) {
         //Toast.makeText(getApplicationContext(), "Tnx Network", Toast.LENGTH_LONG).show();
         locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 0, 2, new LocationListener() {
            @Override
            public void onLocationChanged(Location location) {
               Toast.makeText(getApplicationContext(), "Tnx for updating..",
            }

            @Override
            public void onStatusChanged(String provider, int status, Bundle extras) {
               locationManager.removeUpdates(this);
            }
```

```
        @Override
        public void onProviderEnabled(String provider) {

        }

        @Override
        public void onProviderDisabled(String provider) {

        }
    });
```

// JSON-Firebase

```
package com.example.codesigood.agency;

import android.util.Log;
import com.google.android.gms.maps.model.LatLng;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

/**
 * Created by anupamchugh on 27/11/15.
 */

public class DirectionsJSONParser {
   /** Receives a JSONObject and returns a list of lists containing latitude and longitude */
   public List<List<HashMap<String,String>>> parse(JSONObject jObject){

      List<List<HashMap<String, String>>> routes = new
ArrayList<List<HashMap<String,String>>>() ;
      JSONArray jRoutes = null;
      JSONArray jLegs = null;
      JSONArray jSteps = null;

      try {

         jRoutes = jObject.getJSONArray("routes");

         /** Traversing all routes */
         for(int i=0;i<jRoutes.length();i++){
            jLegs = ( (JSONObject)jRoutes.get(i)).getJSONArray("legs");
            List path = new ArrayList<HashMap<String, String>>();
```

```
 * Method to decode polyline points
    * Courtesy : http://jeffreysambells.com/2010/05/27/decoding-polylines-from-google-maps-direction-api-with-java
    * */
   private List decodePoly(String encoded) {

       List poly = new ArrayList();
       int index = 0, len = encoded.length();
       int lat = 0, lng = 0;

       while (index < len) {
          int b, shift = 0, result = 0;
          do {
             b = encoded.charAt(index++) - 63;
             result |= (b & 0x1f) << shift;
             shift += 5;
          } while (b >= 0x20);
          int dlat = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
          lat += dlat;

          shift = 0;
          result = 0;
          do {
             b = encoded.charAt(index++) - 63;
             result |= (b & 0x1f) << shift;
             shift += 5;
          } while (b >= 0x20);
          int dlng = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
          lng += dlng;

          LatLng p = new LatLng((((double) lat / 1E5)),
                (((double) lng / 1E5)));
          poly.add(p);
       }

       return poly;
   }
}
```