UNIVERSITY OF NAIROBI

SCHOOL OF COMPUTING AND INFORMATICS.

AN ARDUINO BASED VEHICLE TRACKING SYSTEM USING GPS AND GSM MODULES.

BRIAN KIBET RONOH.

P15/81773/2017.

SUPERVISOR: DR. ANDREW MWAURA KAHONGE

May, 2021

A project documentation submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science, School of Computing and Informatics, University of Nairobi

DECLARATION

I declare that this project report and accompanying project implementation, submitted to the School of Computing & Informatics, College of Biological and Physical Sciences, University of Nairobi, for the award of a degree in Bachelor of Science, Computer Science, is my original work and has to the best of my knowledge, not been submitted to any other institution of higher learning for the award of this or any other degree.

Signed	Date
B.K.R	May 19, 2021

Brian Kibet Ronoh – P15/81773/2017.

This project has been submitted in partial fulfillment of the requirements of the Bachelor of Science in Computer Science of the University of Nairobi with my approval as the university supervisor.

Signed	Date

Dr. Andrew Mwaura Kahonge.

ACKNOWLEDGEMENT

I thank the Lord God Almighty for giving me the strength and good health to be able to complete this project with success.

My sincere gratitude to my supervisor, Dr. Andrew Mwaura Kahonge, for his support, technical guidance and encouragement during the course of the project. His valuable guidance has been the one that helped me patch this project and make it full proof. Thank you Dr. Kahonge.

In addition, I would like to express my sincere gratitude to my friends, Hashim Athman and Lewis Kimani, who helped analyze my idea and gave vital insights that helped me out. Without your help, this project would not be complete.

Abstract.

Automobiles are necessary for the movement of goods from one location to another. Consumers may face several problems as a result of delays in the delivery of goods. This delay may be due to drivers choosing incorrect or longer routes when delivering. To avoid these challenges, the Global Positioning System (GPS) is increasingly being used for management of vehicle fleets, recovery of stolen vehicles, mapping and surveillance. This project outlines the design and implementation of a real time GPS tracker system using Arduino. This proposal has significant application for vehicle security, salesman tracking, car hire businesses and private drivers.

An efficient vehicle tracking system is designed and implemented for tracking the movement of any equipped vehicle from any location at any time. The proposed system makes good use of popular technology that combines a smartphone with an Arduino UNO. This is easy to make and inexpensive as compared to others. This system works using the Global Positioning System (GPS) and Global System for Mobile Communication (GSM) technology that is one of the most common ways for vehicle tracking. The device is embedded inside a vehicle those positions is to be determined and tracked in real time. An Arduino UNO is used to control the GPS receiver and GSM module. The vehicle tracking system uses the GPS module to get geographic coordinates at regular time interval. The GSM module is used to transmit and update the vehicle location to a database. This paper gives minute by minute update about vehicle location by sending SMS through GSM modem. This SMS contain latitude and longitude of the location of vehicle. Arduino UNO gets the coordinates from GPS modem and then it sends this information to user in text SMS. GSM modem is used to send this information via SMS sent to the owner of the vehicle. Location is displayed on LCD. And then Google map displays location and name of the place on cell phone. Thus, the user is able to continuously monitor a moving vehicle on demand using a smartphone and determine the estimated distance and time for the vehicle to arrive at a given destination.

Table of Contents

List of Figures.

List of Tables.

Abbreviations.

GPS	- Global Positioning System
GSM	- Global System for Mobile Communication
SMS	- Short Message Service
AVL	- Automatic Vehicle Location
GIS	- Geographic Information System
RF	- Radio Frequency
AGPS	- Assisted Global Positioning System
RFID	- Radio Frequency identification
API	- Application Programming Interface
DC	- Direct Current
IDE	- Integrated Development Environment

CHAPTER 1: INTRODUCTION

Project overview.

With advancements in technology, there has been an increase in the usage of vehicle tracking systems. Security, especially theft, of vehicles in common parking places has become a matter of concern. Commercial fleet operators are by far the largest users of vehicle tracking systems. These systems are used for operational functions such as routing, security, dispatch and collecting on board information. Tracking systems have found applications in areas such as military, navigation, automobiles, aircrafts, fleet management, remote monitoring, remote control, security systems, tele services, etc.

Previous approaches to vehicle tracking use expensive microcontrollers. Deploying a GPS based vehicle tracking system for a small company is still a nightmare when it compares with the setup and the running costs involved in such deployment. Most of the currently available GPS base vehicles tracking systems are satellite based and are very costly, and thus it is not affordable to many. Satellite based tracking systems are not very much affordable and cannot be commonly deployed by most clusters in the developing region.

The proposed system is an embedded system, which is used to know the location of the vehicle using the popular and readily available technologies like the Global Positioning System (GPS) and Global System for mobile communication (GSM). The main feature of our design is that it is proposed to use a development board, which will have GPS and GSM module not as a separated module but closely linked with a microcontroller as in Arduino Uno R3. The advantage of using this development board is that it will reduce the size of whole system and it will reduce the power loss in terms of heat through external wirings used for the connection of GPS and GSM module with the microcontroller. Along with that, it will also increase the durability of the entire system. The Arduino Uno microcontroller will provide the interfacing to various hardware peripherals. To know the location of vehicle, the mobile user has to log in to the Android app and the landing page automatically displays the vehicle’s current location on a base map. Alternatively, for users without an internet connection, they can request for the location via SMS by sending the word “GPS”. A reply message will be automatically sent through the SIM present in the GSM module present in the device. The message will contain the coordinates (sensed by the GPS module) of the vehicle on the registered mobile user and a link to Google Maps which will plot these coordinates on Google Maps.

1. Problem Statement.

The number of vehicles are drastically increasing, and security flaws is becoming a major issue. The global issue related to a constantly increasing crime rate needs to be urgently addressed by both developed and developing countries. In 2018, private car theft rate for Kenya was 0.9 cases per 100,000 population (1370 cases). Though Kenya private car theft rate fluctuated substantially in recent years, it tended to decrease through 2004 - 2018 period ending at 0.9 cases per 100,000 population in 2018 (Kenya Private car theft rate, 2003-2020 - knoema.com, 2021). The figure doesn't account for unreported stolen vehicles – many owners don't report stolen cars as they don't believe they'll get them back. If not recovered soon, stolen vehicles are generally sold, revamped or even burned if the resale price is considered to be too low. Once a vehicle is stolen, it becomes hard to locate it and track it, which considerably decreases the chances of recovering it. Hence, there is need for fleet-based companies, individuals and companies to use a tracking system to track and trace their cars in case of theft or monitor misuse by employees. In this work, I propose the design and implementation of a car tracking anti-theft system that will protect and secure vehicles.

Project Objectives.

i. Research objectives.

1) Research on the types of tracking systems
2) Research on the technologies used in contemporary vehicle tracking systems.
3) Research on How GSM and GPS technologies are used in tracking systems.
4) Research on NMEA data format used in GPS tracking.
5) Research on the use of vehicle tracking software in Kenya and examples of tracking systems already being used in the automotive sector.

ii. System Development Objectives.

1) Gather requirements of the system and come up with a requirement specification document for analysis.
2) Design the vehicle tracking system, including graphical user interface (GUI) design of the Android app and hardware circuits.
3) Implement the detailed design of the system
4) Test and evaluate the implemented system.

In order to fulfil the stated objectives several steps must be taken. These steps involve both software programming and hardware implementation.

These steps are as follows:

1) Establishing a wireless network communication between the GSM module and the smartphone, using a microcontroller (Arduino-Uno).
2) Create a simple yet reliable vehicle tracking system using Arduino-Uno as a microcontroller that will be the medium between the GPS and the GSM module so that the embedded system works efficiently.
3) To find a suitable place locator app (in this project I will be using Google Maps) that will work efficiently with the internet connection (online as well as offline) in order to track the vehicles.
4) Program the Arduino-Uno board in a way that will let it interact with the GPS and GSM module directly and easily.

The main outputs from the expected vehicle tracking system to the end-user are summarized as follows:

1. Plot tracking vehicle's current location on Google Maps.
2. Send location data with a link as an SMS to the user.

Constraints.

1. Cost/Budget.

The project uses hardware and thus budgetary limitations exist. An Arduino Uno R3 microcontroller is used as it is inexpensive and readily available. The GPS and GSM modules interfaced with the Arduino Uno microcontroller are also costly. The budget for this project is estimated to be Kes 10,000.

2. The target hardware.

The Arduino Uno microcontroller has little functionality unlike the Raspberry Pi microcontroller. It does not have an inbuilt GPS module unlike the Raspberry Pi processor. The workaround to this will be to acquire separate modules to interface with the Arduino Uno microcontroller.

3. Limitations on access to specific resources.

Access to GPS and GSM modules are limited as only a few sellers in Kenya deal with the products.

4. The research topic.

Resources on real-time vehicle tracking using cheap micro-controllers e.g., Arduino UNO are limited hence the need for further research on the topic.

Research topics.

1. The types of vehicle tracking systems.
2. The technologies used in contemporary vehicle tracking systems
3. How GSM and GPS technologies are used in tracking systems.
4. NMEA data format used in GPS tracking.
5. The use of vehicle tracking software in Kenya and examples of tracking systems already being used in the automotive sector.

System Context Diagram.

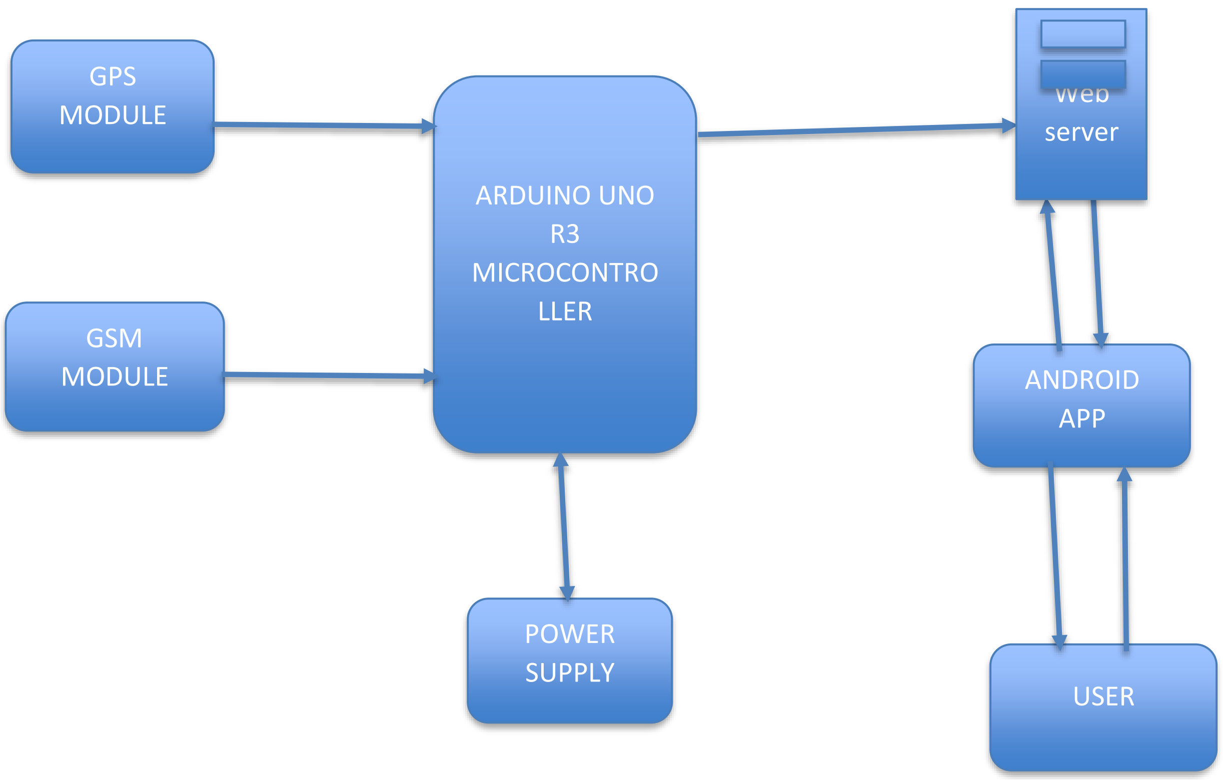


Figure 1: System context diagram.

CHAPTER 2: LITERATURE REVIEW.

Vehicle tracking systems.

A vehicle tracking system combines the installation of an electronic device in a vehicle, or fleet of vehicles, with purpose-designed computer software at least at one operational base to enable the owner or a third party to track the vehicle's location, collecting data in the process from the field and deliver it to the base of operation. Modern vehicle tracking systems commonly use GPS or GLONASS technology for locating the vehicle, but other types of automatic vehicle location technology can also be used. Vehicle information can be viewed on electronic maps via the Internet or specialized software. Urban public transit authorities are an increasingly common user of vehicle tracking systems, particularly in large cities.

Several types of vehicle tracking devices exist. Typically, they are classified as either "passive" or “active".

Active versus Passive Tracking.

"Passive" devices store GPS location and maybe other information such as speed, heading and sometimes a trigger event such as key on or off, door open or closed. Once the vehicle returns to a predetermined point, the device is removed and the data downloaded to a computer for evaluation. Passive systems include auto download type that transfer data via wireless download.

"Active" devices also collect the same information but usually transmit the data in real-time via cellular or satellite networks to a computer or data centre for evaluation.

Passive trackers do not monitor movement in real-time. When using a passive GPS tracker, you will not be able to follow every last move that a tracked person or object makes. Instead, information that is stored inside of a passive tracker must be downloaded to a computer. Once tracking details have been downloaded, it is then possible to view tracking details. After we have gathered all of the information we need from a passive tracker, we can place the tracker back on the same (or different) vehicle. Aside from the fact that a passive tracking device is entirely reliable, the main reason people choose passive trackers is that these devices are less expensive than active trackers. Most passive GPS tracking devices are not attached to a monthly fee, which makes these trackers affordable.

On the other hand, active GPS trackers will allow one to view tracking data in real-time. As soon as an active tracker is placed on a vehicle, one is able to view location and other information such as stop duration and speed from the comfort of their homes or offices. Active GPS trackers are ideal when it comes to monitoring vehicles that need to be tracked at a regular time interval.

There are many advantages associated with a real time tracker. The most important advantage is the convenience of the tracker. Rather than waiting to download data to a computer (as is the case with most passive trackers), a tracker that works in real-time does not require any waiting. Since real-time trackers come with software that allows a user to track an object in real-time, watching any object's progress is simply a matter of sitting at a computer.

Many modern vehicle tracking devices combine both active and passive tracking abilities: when a cellular network is available and a tracking device is connected it transmits data to a server; when a network is not available the device stores data in internal memory and will transmit stored data to the server later when the network becomes available again. Historically vehicle tracking has been accomplished by installing a box into the vehicle, either self-powered with a battery or wired into the vehicle's power system. For detailed vehicle locating and tracking this is still the predominant method; however, many companies are increasingly interested in the emerging cell phone technologies that provide tracking of multiple entities.

Types of Tracking Systems.

There are three main types of GPS vehicle tracking that are widely used. They all use active devices. They are:

1. Automatic Vehicle Location (AVL) system 2.
2. Assisted Global Positioning System (AGPS) 3.
3. Radio Frequency Identification (RFID)

Automatic Vehicle Location (AVL) system

AVL system is an advanced method to track and monitor any remote vehicle with the device that receives and sends signals through GPS satellites. AVL comprises of Global Positioning System (GPS) and Geographic Information System (GIS) in order to provide the real geographic location of the vehicle. AVL system consists of PC-based tracking software to dispatch, a radio system, GPS receiver on the vehicle and GPS satellites. Among the two types of AVL, GPS-based and Signpost-based, GPS-based system is widely used. The tracking method uses GPS satellite to locate the vehicle equipped with GPS modem by sending satellite signals. The accuracy of the tracking method depends on the AVL system which provides the vehicle location with the accuracy of about 5m to 10m. The information transmitted by the tracking system to the base station is location, speed, direction, mileage, start and stop information and status of vehicle. The information of the vehicle is often transmitted to the central control system (base station) from the vehicle after every 60 seconds. If the base station receives the data, it displays it on a computerized map. GPS receiver on the vehicle receives the signals of its geographic location. If AVL system is used to track a vehicle the average cost of per vehicle is $1 to $2 per day.

The system can provide additional services like: vehicle route replay facility, external sensor data, speed alerts. The system also has some limitations; using the AVL system we cannot get accurate, complete and sufficient satellite data in dense urban areas or indoors and when transmission is blocked by natural obstructions (heavy tree cover) or many buildings. It can also occur in RF-shadowed environments and under unfriendly Radio Frequency (RF) conditions. Sometimes, a position fix can be impossible.

Assisted GPS (AGPS) system

In AGPS system, a terrestrial RF network is used to improve the performance of GPS receivers as it provides information about the satellite constellation directly to the GPS receivers. AGPS uses both mobiles and cellular networks to locate the accurate positioning information. AGPS is used to overcome some limitations of GPS. With unassisted GPS, locating the satellites, receiving the data and confirming the exact position may take several minutes. AGPS uses GPS satellites to track the vehicles. A GPS receiver in the vehicle is always in contact with 4 satellites (3 satellites determine latitude, longitude and elevation and the fourth provides an element of time) hence it never fails to detect the location of a vehicle. Location of the vehicle is provided with accuracy of between 3m and 8m, and speed of 1km using this method. Information like Vehicle location, average speed, direction, path traversed in a selected period and alerts (Engaged/Unengaged, speed limit, vehicle breakdown and traffic jam) are delivered by the tracking system to the base station. The system provides continuous updates after every 10 seconds while the vehicle is in motion. It also provides data storage for up to 1 year. The location is retrieved from the GPS device and relayed as a SMS using the cell phone by the Client Node to the Base station. This system is more expensive than the AVL system as it gives a continuous update of the vehicle location. If the user needs an update after every 10 seconds then the subscription for this system is $1.33 per day per vehicle and if the user needs an update after every 5 seconds it is $1.67 per day per vehicle. The system can provide further services like atomic time (Accurate Time Assistance). There is a "panic" button. When pressed, you can contact an operator and he or she will help you out or keep you safe from accidents or hijacks. The system has also some limitations as GSM network is used to transmit data from the vehicle to the base station, and the cost of sending SMS is a major concern to be considered.

Radio Frequency Identification (RFID) System

RFID is an automatic identification method using devices called tags to store and remotely retrieve data. RFID uses radio waves to capture data from tags. The tracking method of RFID comprises of three components: tag (passive, semi passive and active), reader (antenna or integrator) and software (middleware). RFID tag which contains microelectronic circuits sends the vehicle information to a remote RFID reader which is then read via the software. This system provides the location of the vehicle with the accuracy of 4m to 6m. Information such as location of the vehicle, mileage and speed are delivered by the tracking system to the centre. The information is updated every one minute. The information is sent to and received from RFID tags by a reader using radio waves. RFID reader, basically a radio frequency (RF) transmitter and receiver, is controlled by a microprocessor or digital signal processor (DSP). RFID reader with an attached antenna reads data from RFID tags.

GPS (Global Positioning systems)

GPS is a satellite-based radionavigation system owned by the United States government and operated by the United States Space Force. It is one of the global navigation satellite systems (GNSS) that provides geolocation and time information to a GPS receiver anywhere on or near the Earth where there is an unobstructed line of sight to four or more GPS satellites (What is GPS? Everyday Mysteries, 2021). Obstacles such as mountains and buildings block the relatively weak GPS signals.

GPS receivers constantly listen for signals from satellites. The GPS receivers calculate the distance from four or more satellites to accurately figure out its location. Current GPS receivers have a high accuracy, pinpointing to within 1 meter.

GPS data is displayed in different message formats over a serial interface. There are standard and non-standard (proprietary) message formats. Nearly all GPS receivers output NMEA data. GPS receiver communication is defined within the NMEA-0183 data format specification.

NMEA-0183 Standard.

NMEA 0183 (or NMEA for short) is a combined electrical and data specification for communication between marine electronic devices such as echo sounder, sonars, anemometer, gyrocompass, autopilot, GPS receivers and many other types of instruments. It has been defined by, and is controlled by, the U.S.-based National Marine Electronics Association. (Understanding NMEA - VMAC GPS/GSM, 2021)

The NMEA 0183 standard uses a simple ASCII, serial communications protocol that defines how data is transmitted in a "sentence" from one "talker" to multiple "listeners" at a time.

GPS receiver communication is defined within this specification. Most computer programs that provide real time position information understand and expect data to be in NMEA format. This data includes the complete PVT (position, velocity, time) solution computed by the GPS receiver. The idea of NMEA is to send a line of data called a sentence that is totally self-contained and independent from other sentences. There are standard sentences for each device category and there is also the ability to define proprietary sentences for use by the individual company. All of the standard sentences have a two-letter prefix that defines the device that uses that sentence type. (For GPS receivers the prefix is GP.) which is followed by a three-letter sequence that defines the sentence contents.

The purpose of NMEA is to give equipment users the ability to mix and match hardware and software. NMEA-formatted GPS data also makes life easier for software developers to write software for a wide variety of GPS receivers instead of having to write a custom interface for each GPS receiver. For example, VisualGPS software (free), accepts NMEA-formatted data from any GPS receiver and graphically displays it (Gakstatter and Gakstatter, 2021). Without a standard such as NMEA, it would be time-consuming and expensive to write and maintain such software.

NMEA Message structure.

To understand the NMEA message structure, we examine a $GPGGA message output from a GPS receiver:

$GPGGA,181908.00,3404.7041778, N,07044.3966270, W,4,13,1.00,495.144, M,29.200, M,0.10,0000*40

All NMEA messages start with the $ character, and each data field is separated by a comma.

- **GP** represent that it is a GPS position (GL would denote GLONASS).
- **181908.00** is the time stamp: UTC time in hours, minutes and seconds.
- **3404.7041778** is the latitude in the DDMM.MMMMM format. Decimal places are variable.
- **N** denotes north latitude.
- **07044.3966270** is the longitude in the DDDMM.MMMMM format. Decimal places are variable.
- **W** denotes west longitude.
- **4** denotes the Quality Indicator:

    - 1 = Uncorrected coordinate
    - 2 = Differentially correct coordinate (e.g., WAAS, DGPS)
    - 4 = RTK (Real Time Kinematic) Fix coordinate (centimeter precision)
    - 5 = RTK Float (decimeter precision.
- **13** denotes number of satellites used in the coordinate.
- **1.0** denotes the HDOP (horizontal dilution of precision).
- **495.144** denotes altitude of the antenna.
- **M** denotes units of altitude (e.g., Meters or Feet)
- **29.200** denotes the geoidal separation (subtract this from the altitude of the antenna to arrive at the Height Above Ellipsoid (HAE).
- **M** denotes the units used by the geoidal separation.
- **1.0** denotes the age of the correction (if any).
- **0000** denotes the correction station ID (if any).
- ***40** denotes the checksum.

The data is separated by commas to make it easier to read and parse by computers and microcontrollers. This data is sent out on the serial port at an interval called the update rate. Most receivers update this information once per second (1Hz), but more advanced receivers are capable multiple updates per second. 5 to 20Hz is possible with modern receivers.

Benefits of vehicle tracking systems to businesses.

Companies are starting to realise the benefits of vehicle tracking systems. The benefits cut across all industries, both in the commercial and public sector. Tracking makes it easier to eliminate fleet inefficiencies such as journey duplication/overlap and unscheduled journeys. It also encourages a safer, more economic driving style among mobile employees and a more efficient call placing. Other benefits include reduced vehicle wear and tear and reduced administration time associated with meeting health and safety policies (Marchet, Perego & Perotti, 2009).

The potential benefits of a vehicle tracking system can be immediate, with enhanced fleet reactivity and productivity making it possible to generate a fast return on investment and increase business capacity. It can also assist with meeting the needs of government legislation and security for mobile employees (Ting, Wang &Ip, 2012).

Vehicle tracking is a way to improve company efficiency and in effect, increase profitability, especially in the business of large vehicle fleets (Hsieh, Yu, Chen & Hu, 2006). Vehicle tracking systems are the enabling technology, and is the key to release the value trapped in asset management. By its non-contact, scan-based data reading characteristics, it automates the asset tracking and data acquisition that enables an enterprise to locate vehicles (cars, trucks, etc.) and even uses location information to optimize services. With the help of tracking information, the manager is able to access one or more driver locations and gets their status information on a real-time basis (for instance, checking if the drivers execute the order; if they follow the driving routes; if there is any traffic congestion (Roh, Kunnathur &Tarafdar, 2009).

Existing Vehicle tracking systems in Kenya.

STrack Kenya LTD.

STrack Kenya Limited is a company founded in the year 2006 to meet national and personal security needs of Kenya and the East Africa Region. The Company specializes in provision of unmatched vehicle tracking and fleet management service delivery for motor vehicles and assets using Global Positioning System (GPS) Technology, supported by proven advanced technology devices from Sweden, an online tracking software based on Google Maps and a Linux server environment for enhanced reliability and user security.

Regent Tracking Services.

Regent Tracking Services Limited is a security solutions provider and a member of Regent Automobile Group. The company provides services ranging from fleet management solutions, security and surveillance to vehicle and asset tracking. Their fleet management solutions use advanced GPS tracking and real-time vehicle monitoring and control.

Myriad Services Ltd (MSL)

Myriad Services Ltd is a company that provides fuel management, fleet monitoring, on board vehicle cameras, vehicle accessories services and many other services. It is a Kenyan company with offices in Nairobi, Mombasa and representatives in Kenya major towns, and specializes in the procurement and supply of Fuel Management solutions and many other services to the private and public sector corporations within East Africa.

Nairobi Car Trackers.

Nairobi Car Trackers is a company providing tailor made automobile tracking and fleet management solutions to various companies in Kenya. Their services include: Vehicle GPS tracking system, motorbike GPS tracking system and an auto watch system that prevents unauthorized access into a vehicle. Key features of their tracking systems are: Stop engine remotely, set speed limit, Geo-fence real time tracking, Satellite view, Traffic monitoring, History playback (3 months), Alerts on Over-speeding, low battery, Engine on/off, external power disconnected, Reports on Mileage, Parking details, Trip report, Alert reports.

Pinnacle Systems (K) Ltd

Key features of their tracking system include: Live, web-based Vehicle Tracking, Vehicle Tracking uses high quality Mapping including Satellite and Street view, Journey Trails and Replays, Find Nearest vehicle and Job Dispatch Tools, Monitoring of Driver behaviour, Points of Interest, Barred Locations and Geo Zones, Comprehensive reporting suited to Ad hoc and Automated Reports.

Smart Track Vehicle Tracking System

Smart Track Vehicle Tracking System is a Real-time vehicle tracking system that relies on both the Global Positioning System (GPS) satellites and a cellular system. It was developed by Endeavor Africa Group. Key software features include: fuel management, tyre management, fleet maintenance, Real Time vehicle location monitoring, Multiple and secure login, SMS/ Email alerts and scheduler, Route planning and analysis, History / Speed analysis, Route deviation notification and Report analysis.

Other tracking systems/companies include: **Rivercross Tracking, Trailmycar, Solutions Unlimited tracking system, Carro Tracking Solutions and Nimba Technologies LTD.**

CHAPTER 3: SYSTEM DEVELOPMENT METHODOLOGY.

System Development Life Cycle (SDLC).

The System Development Life Cycle (SDLC) is a conceptual model for software development that consists of multiple phases: Software Conceptualization; Analysis; Design; Coding and Debugging; System Integration and Testing; Implementation; Maintenance and Support. Each phase can be thought of as a building block for the next phase.

There are different SDLC models that may be followed, such as the classic “Waterfall Model,” “Spiral,” and “Evolutionary Prototyping,” as well as many modified Waterfall models.

The waterfall approach is one of the oldest SDLC models, but it has fallen out of favour in recent years. This model involves a rigid structure that demands all system requirements be defined at the very start of a project. Only then can the design and development stages begin.

Once development is complete, the product is tested against the initial requirements and rework is assigned. Companies in the software industry typically need more flexibility than what the waterfall methodology offers, but it still remains a strong solution for certain types of projects, especially government contractors.

The sequential phases in the Waterfall model are:

- **Requirement Gathering and analysis** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- **System Design** − The requirement specifications from the first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
- **Implementation** − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.

- **Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released. Also, to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model, phases do not overlap.

In a practical software development project, the classical waterfall model is hard to use. So, Iterative waterfall model can be thought of as incorporating the necessary changes to the classical waterfall model to make it usable in practical software development projects. It is almost same as the classical waterfall model except some changes are made to increase the efficiency of the software development.

The iterative waterfall model provides feedback paths from every phase to its preceding phases, which is the main difference from the classical waterfall model.

Feedback paths introduced by the iterative waterfall model are shown in the figure below. When errors are detected at some later phase, these feedback paths allow correcting errors committed by programmers during some phase. The feedback paths allow the phase to be reworked in which errors are committed and these changes are reflected in the later phases. But there is no feedback path to the stage – feasibility study, because once a project has been taken, does not give up the project easily.

It is good to detect errors in the same phase in which they are committed. It reduces the effort and time required to correct the errors.

The development of this system uses the iterative waterfall method. The stages of system development process using the iterative waterfall method are illustrated in the following diagram:

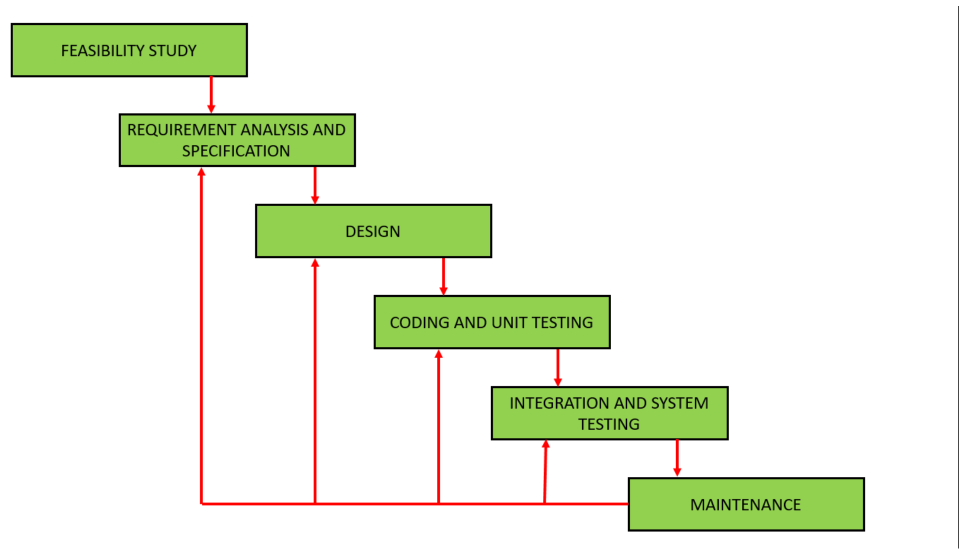

FEASIBILITY STUDY
REQUIREMENT ANALYSIS AND SPECIFICATION
DESIGN
CODING AND UNIT TESTING
INTEGRATION AND SYSTEM TESTING
MAINTENANCE

Figure 2: Steps of the iterative waterfall SDLC Model. From (Software Engineering | Iterative Waterfall Model - GeeksforGeeks, 2021)

This model will be used because of the following reasons: Requirements are very well documented, clear and fixed, the technology is understood and is not dynamic, there are no ambiguous requirements, ample resources with required expertise are available to support this project, clearly defined stages, easy to arrange tasks, Phases are processed and completed one at a time, and it's easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process during the development of the project, the following activities will be carried out at each stage:

Requirements Gathering and analysis.

Activities to be carried out in this phase are as follows:

I. Conduct a feasibility study.
II. Gather requirements using methods such as interviews, questionnaires, internet sources, existing systems analysis and observation of main user groups.
III. Identifying actors.
IV. Identifying scenarios.
V. Identifying use cases.
VI. Refining of use cases.
VII. Identifying relationships among use cases and actors.
VIII. Identifying initial analysis objectives.
IX. Identifying non-functional requirements.
X. Managing requirements elicitation -
XI. Maintaining traceability by documenting the requirements elicitation i.e., coming up with a Requirements Analysis Document (RAD)

The **key deliverable** for this stage is the Requirements Specification document.

System Design.

Objective:

The objective of this phase is to transform business requirements identified during the requirements phase, into a detailed system architecture which is feasible, robust and brings value to the organization.

Activities to be carried out in this phase are:

1) Review of the Requirements elicitation document.
2) Identify the Inputs, outputs, databases, forms, codification schemes and processing specifications of the system.
3) Decide on the programming language and the hardware and software platform in which the new system will run.
4) Develop the system architecture.
5) Develop the master plan for testing and evaluation.
6) Develop the user interface designs.
7) Come up with a database design.

Key deliverables: Flowcharts, Data flow diagram (DFD), Data dictionary.

Implementation.

Objective:

The objective of the Implementation Phase is: first to install the system in the production environment and to bring it into operation; and second, to ensure that the system, as developed:

- Satisfies the functional requirements
- Satisfies the business needs;
- Adheres to all mandates, physical constraints and service level agreements; and
- Operates as described in the requirements phase.

Activities:

1) Review of the design.
2) Selection of standards, methods, and tools for developing the system.
3) Coding - Includes implementation of the design specified in the design document into executable programming language code. The output of the coding phase is the source code for the software that acts as input to the testing and maintenance phase.
4) Document the system by coming up with system documentation.
5) Evaluate and review the final system to ensure it meets the requirements and it maps the design well.

Key deliverable: Source code that is an input to the testing phase and system documentation.

Integration and Testing.

All the units developed in the implementation phase will be integrated into a system after testing of each unit. Post integration the entire system will be tested for any faults and failures.

Evaluation

Once the system is operational, it will be assessed. The system will be given to outside users to test the system and give feedback based on their actions and experience in form of questionnaires. This will help in handling the residual errors that may exist in the software even after the testing phase. This phase will also monitor system performance, rectify bugs and requested changes are made.

Software and hardware requirements.

Hardware requirements.

For designing this system, many types of devices are used to make it perfectly working. The following list of hardware are required for this system.

I. Arduino Uno R3
II. SIM808 Module
III. GPS and GSM antennae
IV. Power Supply - to supply power to the microcontroller board and all other components connected to it.
V. Connecting Wires that connect the various hardware components.
VI. A laptop.

Arduino Uno R3 microcontroller.

The microcontroller used for this project will be Arduino Uno R3. The R3 is the third, and latest, revision of the Arduino Uno. The Arduino Uno is a microcontroller board based on the ATmega328. The ATmega328 has 32 KB (with 0.5 KB occupied by the boot loader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library). It has 20 digital input/output pins (of which 6 can be used as PWM outputs and 6 can be used as analog inputs), a USB connection, a power jack, an in-circuit system programming (ICSP) header, and a reset button. It is simply connected to a computer with a USB cable. The V_{in} is the input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). The 5V pin outputs a regulated 5V from the regulator on the board. The microcontroller board can be supplied with power either from the DC

power jack (7 - 12V), the USB connector (5V), or the V_{in} pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. So, it is advised not to do so. Maximum current draw is 50 mA [5]. An Arduino board is based on an AVR microcontroller chip and when the board with nothing wired or attached to it consumes around 80mA of 5 volts current. The Clock speed of the Arduino is 16 MHz so it can perform a particular task faster than the other processor or controller. The AVR chip is clocking at 16 MHz continuously no matter what the code is doing, it never 'halts' so its current consumption is basically independent of the code you have it execute. Only if you put the AVR chip into one of its 'sleep modes' can you halt code execution and drastically cut current consumption for the AVR chip, however the rest of the other components on the Uno will continue to draw their normal current consumption. Also, the Arduino does not provide any 'sleep mode' examples so one will have to look for other user supplied coding example. Arduino board supports I2C and SPI communication. The Arduino software includes wire library for I2C and SPI library for the SPI communication
Software Components required.

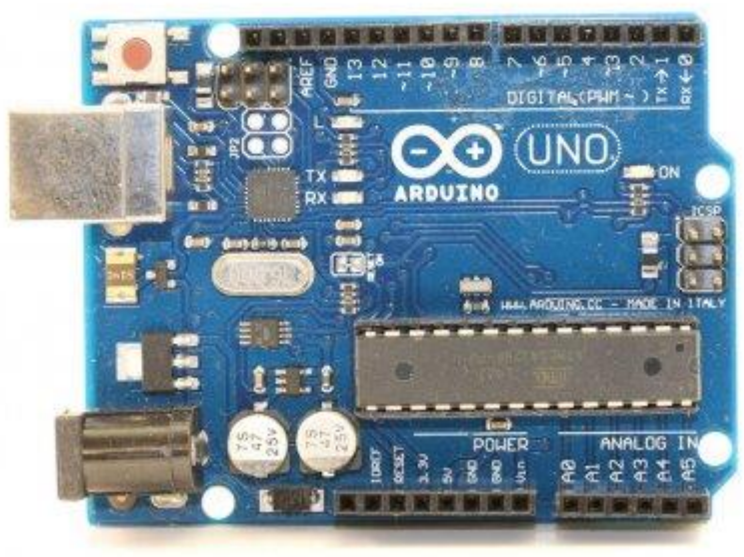


Figure 3: Arduino Uno microcontroller. From (Arduino Uno Rev3, 2021)

SIM 808 Module.

SIM808 module is a complete Quad-Band GSM / GPRS module which combines GPS technology for satellite navigation. It has a SIM application toolkit where SIM card can be inserted. The

compact design which integrated GPRS and GPS in a SMT package significantly saves both time and cost for one to develop GPS enabled applications. A modem GSM & GPRS with SIM808 module allows to create data connections on the GSM network through a standard USB interface. The cellular modems, particularly USB-stick ones, are now at very affordable prices. However, they are limited: they are explicitly designed for Internet connections, so one cannot use it as a normal modem and so implement, for example, a point-to-point data communication with them.
To switch ON the cellular module, the microcontroller has to put high the line ON/OFF (pin 1 on connector). This saturates the T2 transistor that drives to low the line PWR of GSM. SIM808 is designed with power saving technique so that the current consumption is as low as 1.0mA in sleep mode (GPS engine is powered down). The range of DC005 voltage input is 5 - 26V, when use the 5V power as the power, it is needed to make sure that the power supply can provide 2A current.
The SIM808 module has two different serial ports on board, one for the cellular section of the module and one for the GPS section. The serial port on cellular allows the full management of SIM808 module, therefore it can be used to configure and communicate with the GPS receiver, in order to call for data about satellite status and geographical positioning and to transfer them to the microcontroller. This is the approach followed in the design of this project.
All the GPS function is controlled by AT command via serial port. This module uses AT command to execute user's desired functions. While using the GPS function, two AT commands are sent to open the GPS function, and the commands are AT+CGPSPWR=1 and AT+CGPSRST=1 respectively; two instructions are used to power GPS and reset GPS. And then, the GPS TTL level interface will send data out and the baud rate is 115200 by default.
The SIM 808 module interfaces with our GSM and GPS antennas and holds the SIM card used to send messages to the user.

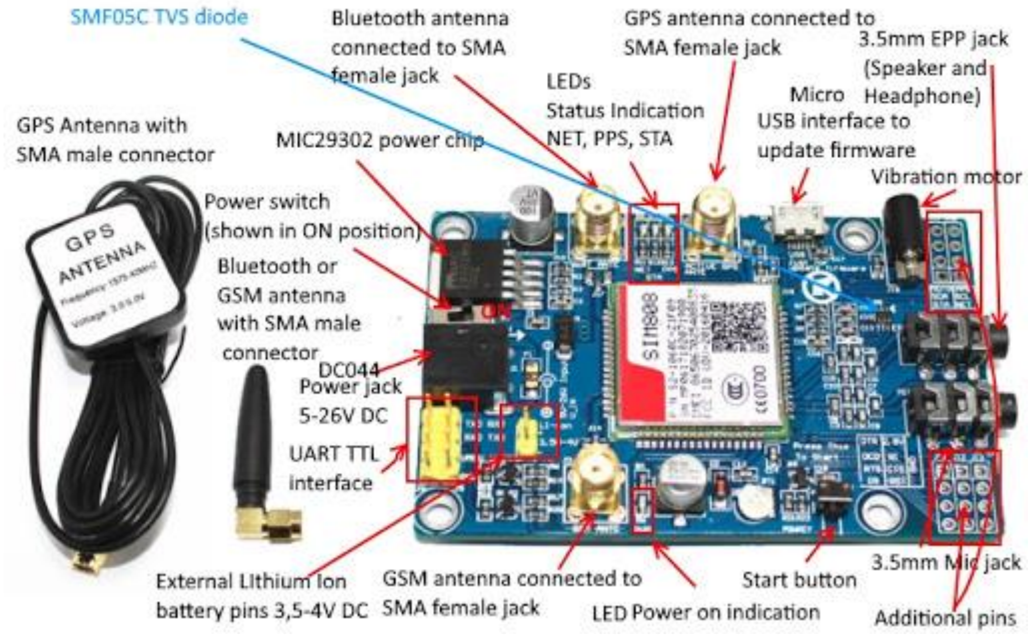

SMF05C TVS diode
Bluetooth antenna connected to SMA female jack
GPS antenna connected to SMA female jack
3.5mm EPP jack (Speaker and Headphone)
LEDs Status Indication NET, PPS, STA
Micro USB interface to update firmware
GPS Antenna with SMA male connector
MIC29302 power chip
Vibration motor
Power switch (shown in ON position)
Bluetooth or GSM antenna with SMA male connector
DC044 Power jack 5-26V DC
UART TTL interface
External Lithium Ion battery pins 3,5-4V DC
GSM antenna connected to SMA female jack
LED Power on indication
Start button
3.5mm Mic jack
Additional pins

Figure 4: SIM 808 front side, Source: (SIM808 GSM/GPRS/GPS Bluetooth Compatible Development Board With GPS Antenna COM45 - Faranux Electronics, 2021)

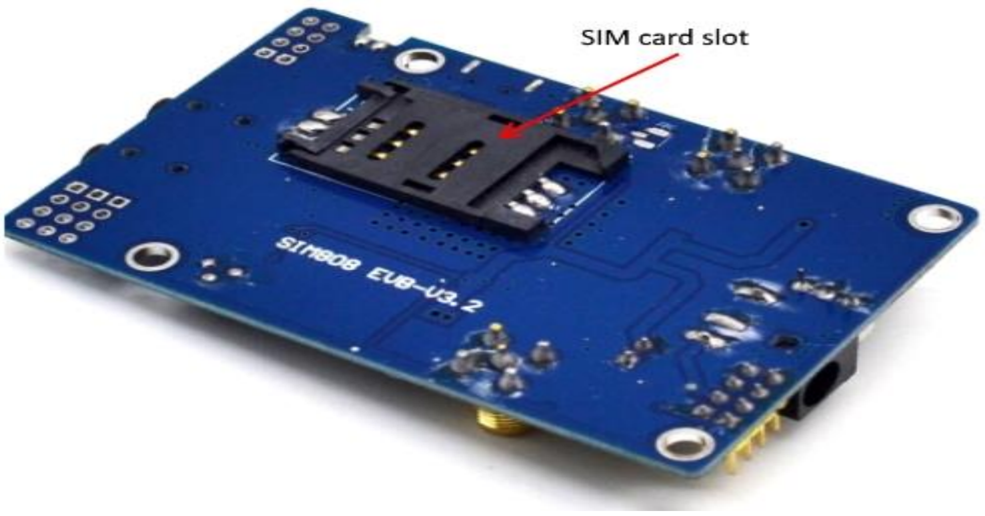


Figure 5: SIM 808 Backside, Source: (Basics: Project 053d SIM808 GSM GPRS GPS Bluetooth evolution board (EVB-V3.2) at Acoptex.com / ACOPTEX.COM, 2021)

GPS and GSM antenna.

GPS antenna.

This GPS antenna draws about 10mA and will give you an additional 28 dB of gain. It’s got a 5-meter-long cable so it will easily reach wherever it is needed to. The antenna is magnetic so it will stick to the top of a car or truck or any other steel structure. Its operating frequency range is 1575.42±1.023 MHz and voltage range is 2.5V- 5.5V and corresponding current range is 6.6 mA - 16.6 mA [9].

GPS signals are extremely weak and present unique demands on the antenna so the choice of antenna plays an important role in GPS performance. A GPS unit needs to have a clear, unobstructed sky view, to best receive the microwave signals that allow it to communicate with satellites. GPS Down/Up converter used for very long cable runs. This GPS antenna that receives the GPS signal, converts it to a lower frequency which is then sent down the cable. Next to the GPS receiver is an up converter that converts the signal back to the original frequency and delivers it to the GPS receiver.

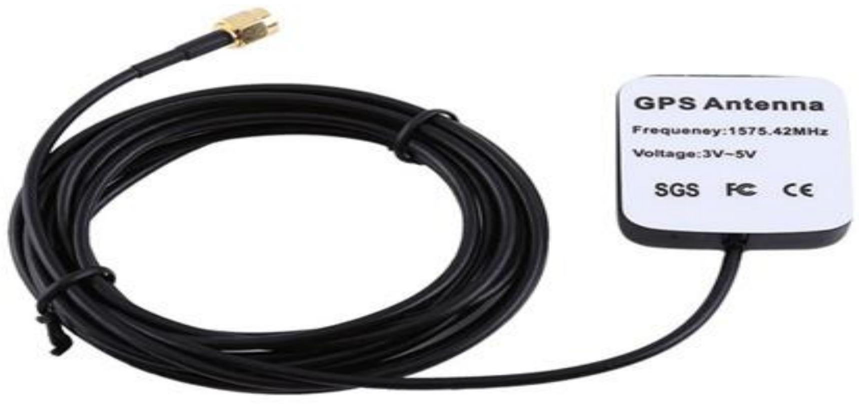


Figure 6: GPS active antenna, Source: (SIM808 EVB-V3.2 Module GSM GPRS GPS Development Board SMA GPS Antenna Arduino, 2021)

GSM antenna.

GSM communications are dependent on antennas. The antenna is what allows communications signals to be sent and received. The antenna that we have used in our project provides operation at both GSM Quad Band Frequencies with +2dBi gain [10]. This antenna operates in Quad Band 890/960, 1710/1880 MHz Frequencies and it’s omni-directional.

Figure 7: GSM antenna, Source: (SIM808 EVB-V3.2 Module GSM GPRS GPS Development Board SMA GPS Antenna Arduino, 2021)

Software Requirements.

Arduino IDE Compiler.

The Arduino IDE is a cross-platform application written in Java, and is derived from the IDE for the Processing programming language and the Wiring project. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and is also capable of compiling and uploading programs to the board with a single click. There is typically no need to edit make files or run programs on a command-line interface. Although building on command-line is possible if required with some third-party tools such a Ino.

The Arduino IDE comes with a C/C++ library called "Wiring" (from the project of the same name), which makes many common input/output operations much easier. Arduino programs are written in C/C++.

This is where code to program our microcontroller is edited, compiled and loaded into the microcontroller.

Google Maps.

Desktop and mobile web mapping service application and technology provided by Google, offering satellite imagery, street maps, and Street View perspectives, as well as functions such as a route planner for travelling by foot, car, bicycle (beta test), or with public transportation. Also supported are maps embedded on third-party websites via the Google Maps API, and a locator for urban businesses and other organizations in numerous countries around the world. Google Maps satellite images are not updated in real time; however, Google adds data to their Primary Database on a regular basis. Google Earth support states that most of the images are no more than 3 years old.

Google Maps will help in decoding location data (latitude and longitude) to place names and help to plot them on the map.

CHAPTER 4: SYSTEM ANALYSIS.

Introduction

The objective of the system analysis phase is to understand the proposed project by ensuring that it supports business requirements and builds a solid foundation for system development. This chapter contains the user requirement specification on the system which has been gathered from users through the different data collection techniques. Subsequently in this section we concentrate on identifying functional and non-functional requirements of the proposed system. Models and other documentation tools are used to visualize and describe the proposed project.

Requirements elicitation.

Requirements are the functional needs that a particular design, product or process aims to satisfy. This stage involves getting the requirements from the users, which include both the vehicle owners and other key actors in the sector. This is a detailed plan of the process of collecting requirements from the users and stakeholders involved:

Goals of the process.

- To find out how how police officers receive information about vehicle theft.
- To find out how police officers and insurance companies recover stolen vehicles.
- To find out how car hire businesses locate their vehicles when they lease the vehicle to a client.
- To find out the issues faced by vehicle owners regarding vehicle security and fears of car thefts.

People to visit and interview.

- Police officers/ Traffic officers/ NTSA officers
- Vehicle owners.
- Car hire companies/individuals.

Methods used to gather requirements

Various methods were used gather the requirements of the system which include

i. Observation of car owners using existing systems to locate their cars.
ii. Informal interviews with car owners and other stakeholders.
iii. Investigative documentaries on car thefts in Kenya.
iv. Web articles.

v. Existing documentation on fleet management and tracking.

Locations to visit.

- Police station/ Highway checkpoint

Data recording.

- Note taking.

A set of questions to be directed to the different stakeholders were developed and analysed to ensure they were not leading or biased.

Requirements analysis.

This step helped in determining the quality of the requirements. It involved identifying whether the requirements found are unclear, incomplete, ambiguous, and contradictory. These issues were resolved before moving to the next step.

The requirements found from vvarious documents, interviews and investigative documentaries have been categorized as follows:

Car owners and drivers.

Car owners are the main stakeholders of this system. Car owners interviewed gave the following needs of a proposed solution to the problem of car theft:

i. Car owners need a way to locate their vehicles in case they are stolen.
ii. Car owners need a way to report easily and promptly to the police and relevant authorities such as NTSA and their insurance companies.
iii. Car owners need a way to get alerts when their vehicle is out of a certain location range.
iv. Car owners need a way to know if their GPS tracking device has been detached from their car and the details of the last known location.

Car hire companies and car hire individuals

Car hire companies and individuals interviewed needed a way to:

i. View the location and vehicle details such as speed and direction where the car is heading to of their cars in real-time so as to closely monitor their customers.
ii. View the location history of their cars after they have been returned to the base station.
iii. Track over speeding clients and warn them.
iv. Get alerts when their vehicles cross a certain location radius.
v. Track multiple vehicles at the same time on the same map.
vi. Report their missing vehicles promptly to the police and relevant authorities.

NTSA/Kenya Police officers

A traffic police officer was interviewed regarding vehicle theft and vehicle inspection on roads. These officers are extreme users of the system and they will interact with the system possibly on rare occasions. These officers receive complaints of vehicle theft and log them in a missing vehicles list that is sent to the general public. The officer interviewed highlighted the need to make the public aware of NTSA hotline numbers and contacts in case of vehicle theft for prompt reporting.

Requirements specifications.

After using the above methods of data collection in the requirements elicitation phase, I highlighted the main user requirements of the system that I would uphold and organize them into two main categories namely:

1. Functional requirements.
2. Non-functional requirements.

Functional requirements.

1) The system should be capable of showing a vehicle's position on a geographical base map.
2) The system should provide a facility to authenticate the tracking user by user-id and the password.
3) The system should provide a facility to scroll the map left, right, bottom and up.
4) The system should enable a user to zoom in or zoom out the base map.
5) The system should be able to send an SMS to the user of the current location of the vehicle on demand.
6) The system should provide a facility to track the location of multiple cars at the same time.
7) The system should provide a facility to maintain location history including past tracking data with the system.
8) The system should provide a facility to rotate the base map left and right.
9) The system should display the speed of the car.
10) The system should display the direction the car is moving.
11) The system should provide a facility to report a stolen car instantly.
12) The system should provide a way that users enter their car’s information when registering.

Non-functional requirements.

Usability requirements.

i. The hardware device (GSM/GPRS modem) which will be used in automobile should be unsophisticated and affordable.
ii. These hardware devices should be tiny and could be easily fixed on tracking vehicles.
iii. While it is in operation, the activity of such car unit should not be identifiable to the person who drives the vehicle.

Reliability Requirements.

The system should be designed for maximum reliability and flexibility. The following reliability requirements were identified:

i. A low-cost reliable tracking device to be installed in the vehicles.
ii. The proposed system will be totally dependent on the GSM network. Therefore, the reliability of those dependant services will be taken into consideration when measuring the reliability of the proposed system.
iii. Expected overall system reliability is 99.0%.
iv. Mean Time between Failures (MTBF) will be four hours.
v. Mean Time to Repair (MTTR) will be two hours.

Performance requirements.

i. The vehicle tracking system's performance characteristics that include specific response times, accuracy and the system performances are:
ii. Offer real-time tracking with minimum latency. Latency will depend on the internet speed as well as other factors. It should be noted that latency is not an issue in the context of the use of this system. Therefore, a latency of 30 seconds is deemed acceptable.
iii. In order to get coordinate to the system accurately, the car unit should operate properly without sending erroneous data.
iv. Response time should be minimal. It should be less than three minutes.
v. The system shall operate properly 24/7 days.
vi. Accuracy of the result should be within the acceptable level. It should be within 500m to 2km radius in the actual scenario.
vii. Dynamics of map should be created with zoom in/out facility.
viii. Cost effectiveness with respect to a same commercial application.

Supportability requirements.

The requirements that will enhance the supportability or maintainability of the vehicle tracking system are:

i. This hardware will require an internet facility to get connected to the web server
ii. SIM with a minimal data plan and SMS plan.
iii. This system should work from any where, when there is a direct connection to the internet from the vehicle unit.

Hardware requirement specifications.

The system will have hardware components i.e., the mobile unit which will be installed in the car. The car unit will compose of a microcontroller, a GSM and a GPS module all connected together to obtain the current location of the vehicle and send the location data to the server and to the user on demand by SMS. The following are the hardware requirements:

i. Since we will be using an external GPS module, the hardware system needs to have a functionality to setup the GPS receiver by installing the needed drivers and set the needed configuration for the receiver to be detected and read by the Arduino Uno Microcontroller.
ii. Since we will be using an external 2G GSM module, the hardware system needs to have a functionality to setup the 2G GSM module by installing the needed drivers and setting the needed configuration for the module to be detected and interfaced to the Arduino Uno Microcontroller.
iii. The GPS receiver connected to a GPS antenna sends raw geolocation data to the dynamic micro-controller. Therefore, the hardware system has to be able to receive and read this geolocation data using geolocation libraries.
iv. Since the read GPS data is raw, it is still not readable. This is why the hardware system needs to reformat this geolocation data to become readable.
v. The system should be able to parse the read GPS data into different variables including longitude, latitude, altitude, speed as well as other geolocation variables.
vi. The hardware system should be able to send the location data obtained to a remote web server for processing.

Constraints.

i. The GPS/GSM module should run on top of an Arduino microcontroller.
ii. The programming languages used for reading and sending GPS data is C and C++.
iii. The system should send geolocation data related to its current position every 10 seconds.
iv. Multi-threading will be used while reading GPS data in order to prevent a buffer overflow.
v. AT commands will be used for the setup and configuration of the GSM and GPS modules.

Feasibility Study.

Feasibility study measures the practicality of the proposed system into adoption. It assesses whether the vehicle tracking system is a viable project that can be used by the auto industry. The goals of this feasibility study are as follows:

- To understand thoroughly all aspects of the vehicle tracking system, concepts, or plans.
- To become aware of any potential problems that could occur while implementing the project.
- To determine if, after considering all significant factors, that the project is viable—that is, worth undertaking

The following are the factors that determine the project feasibility;

Technical feasibility.

This assessment focuses on the technical resources available to implement the project. It helps to determine whether the technical resources meet the required capacity and whether the technical team is capable of converting the ideas into working systems.

To develop the system, the following software is required:

a) Operating System – Windows 10.
b) Visual Studio Code (Front End Debugger)
c) Arduino IDE
d) Android studio.

Hardware Requirements include:

a) Processor: intel core i5 and above
b) Hard Disk: 500GB and above
c) RAM: 4GB and above
d) Laptop
e) Arduino Uno Microcontroller.
f) SIM808 GPS/GSM module
g) GPS antenna
h) GSM antenna
i) Male-Female Jumper wires.
j) Power adapter.

Most of the software to be used in the development of the system are open source and readily available. However, some hardware resources such as Arduino Uno and GSM/GPS module have to be bought. Documentation of the above software and hardware resources are readily available.

Operational feasibility.

Operational feasibility studies help to describe how well the system will work when implemented and rolled out to the public. The system will not only help vehicle owners monitor their vehicles in real time but also help them to easily and promptly report vehicle thefts to local authorities. This is evident that the system will reduce vehicle thefts.

Schedule feasibility.

This helped review how reasonable the timeline for the proposed system is. The project schedule indicates all the tasks and deliverables throughout the proposed system’s development. The provided period of development is sufficient. The project has a good schedule with every task given enough time to be completed.

Economic feasibility.

This helps to measure the cost-effectiveness of the proposed system in terms of building the system and the returns after the system is complete and functional. This particular project requires minimum funds since all the technical hardware resources are cheap and available while most of the software requirements are free.

Modelling.

Modelling involves graphical methods and non-technical language that represent the system at different stages of development. Various tools are used to describe business processes, requirements and user interaction with the system. Different models like data flow diagrams and Unified Modelling Language diagrams can be used for modelling.

To help give the detailed elaboration of the Vehicle tracking system being analysed, the analysis models below were used;

1. **Functional Decomposition Diagram (FDD)**
2. **Context Diagram**
3. **Data Flow Diagram (DFD)**
4. **Entity Relationship Diagram (ERD)**
5. **Use Cases**

1. Functional Decomposition Diagram (FDD)

Functional decomposition is a method of analysis that dissects a complex process in order to examine its individual elements. A function, in this context, is a task in a larger process whereby decomposition breaks down that process into smaller, easier to comprehend units.

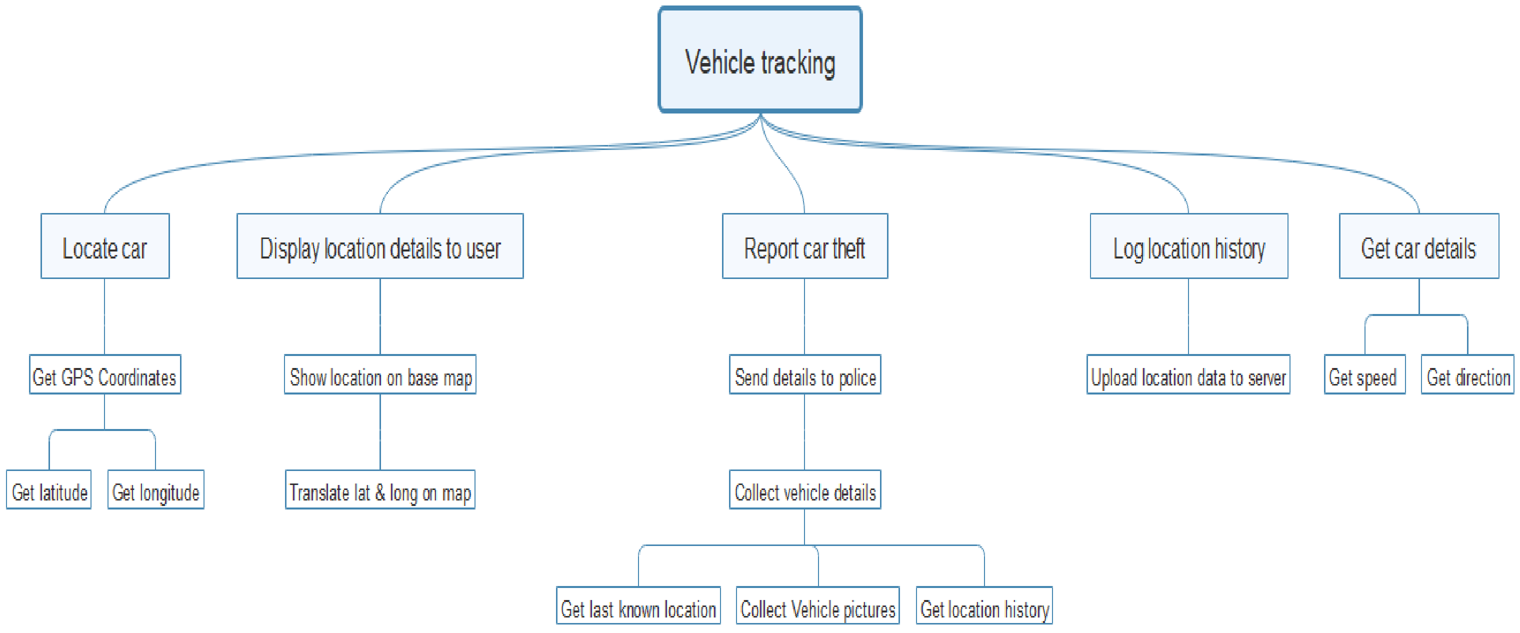
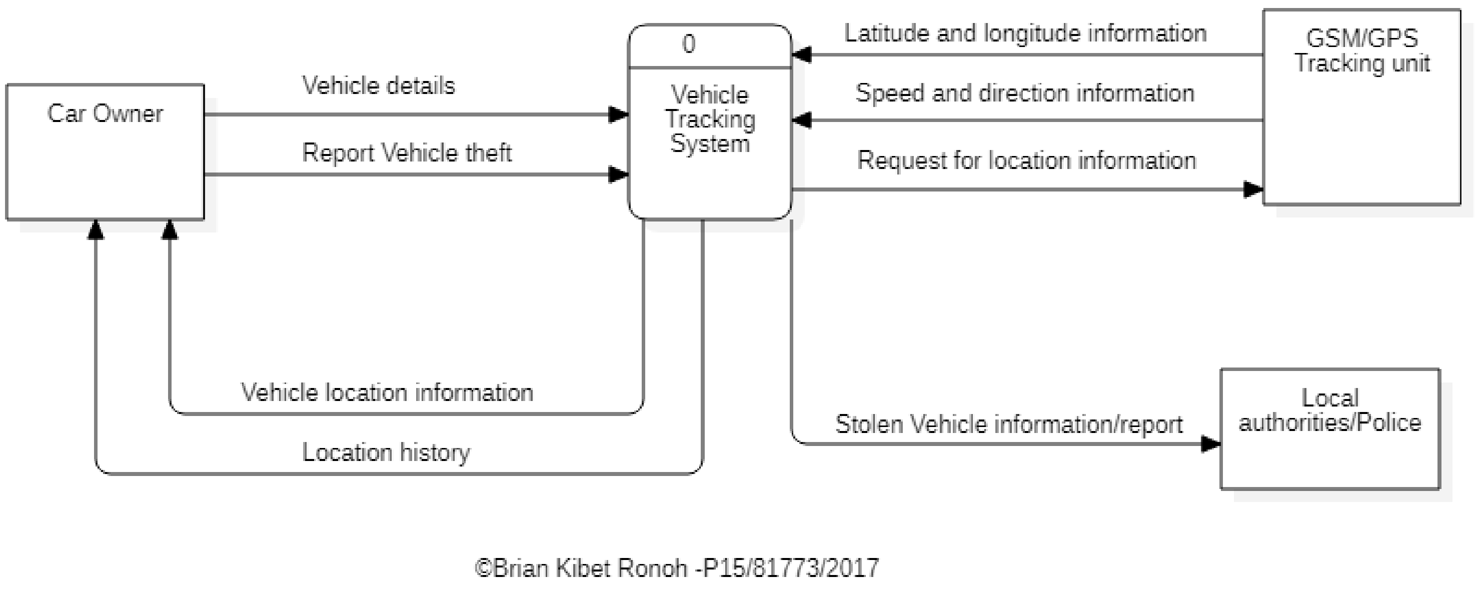


Figure 8: Functional decomposition diagram

2. Context diagram.

The System Context Diagram (SCD) below shows the system as a whole, and its inputs and outputs from/to external factors. This diagram is the highest-level view of a system. It shows the borders, actors, and systems interacting with the vehicle tracking system and the major information flow in and out of the system.



Car Owner
Vehicle details
Report Vehicle theft
0
Vehicle Tracking System
Latitude and longitude information
Speed and direction information
Request for location information
GSM/GPS Tracking unit
Vehicle location information
Location history
Stolen Vehicle information/report
Local authorities/Police
©Brian Kibet Ronoh -P15/81773/2017

Figure 9: Context Flow diagram

The system interfaces with three entities:

1) **Car Owner** – The owner/ driver of the vehicle enters the vehicle details and on installation of the tracking device on the car, they receive the current location information including speed and direction. The car owner can also report theft of the vehicle to the police.
2) **GSM/GPS tracking unit** - Senses the location of the vehicle and report back to the owner.
3) **Local authorities/police** – they receive alerts on vehicles stolen from the system when the owner reports one.

3. Data Flow Diagram.

The data flow diagram below depicts the flow of data within the system and the processing performed by the system.

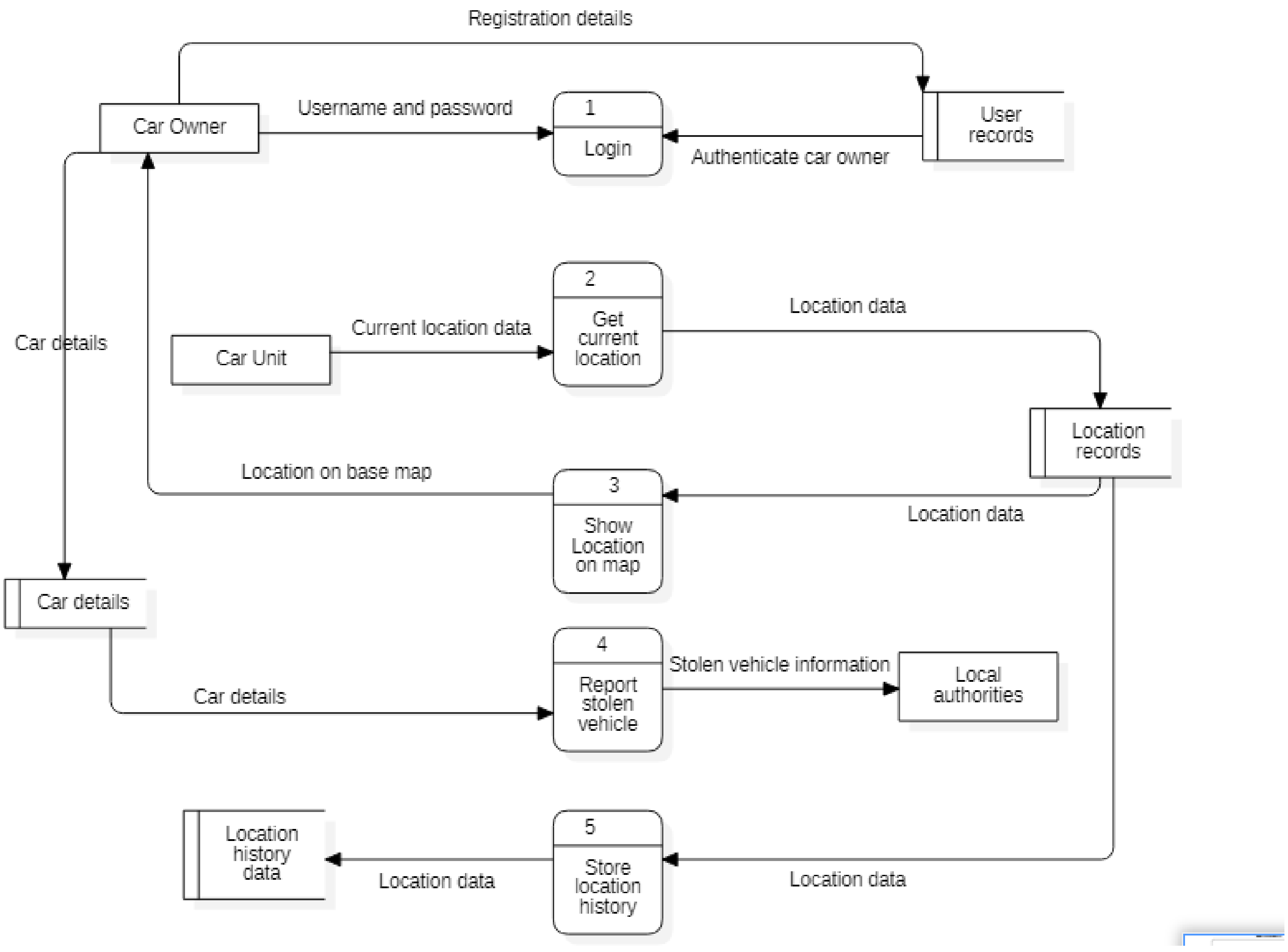


Figure 10: Level 1 Data Flow Diagram

Process 1: Login

The user gives their details to the system i.e., username and password and the system checks from user records (which are obtained when the user registers into the system) to authenticate the user.

Process 2: Get current location.

The car unit senses GPS geolocation data i.e., latitude and longitude and sends the geolocation data to a web server containing location records.

Process 3: Show location on map

This process entails translating location latitude and longitude into human readable format and displaying the exact location on a base map.

Process 4: Report Vehicle theft

This process entails alerting local authorities of a stolen vehicle by sending the last known location of the stolen vehicle and the vehicle details. Data from the location records and car details are fetched and sent to local authorities.

Process 5: Store location history

The system fetches the location data from current location records and logs them into a location history database server

4. Entity Relationship diagram.

The entity relationship diagram has been used to outline representation of the various entities that will exist within the tables and their relationships to each other.

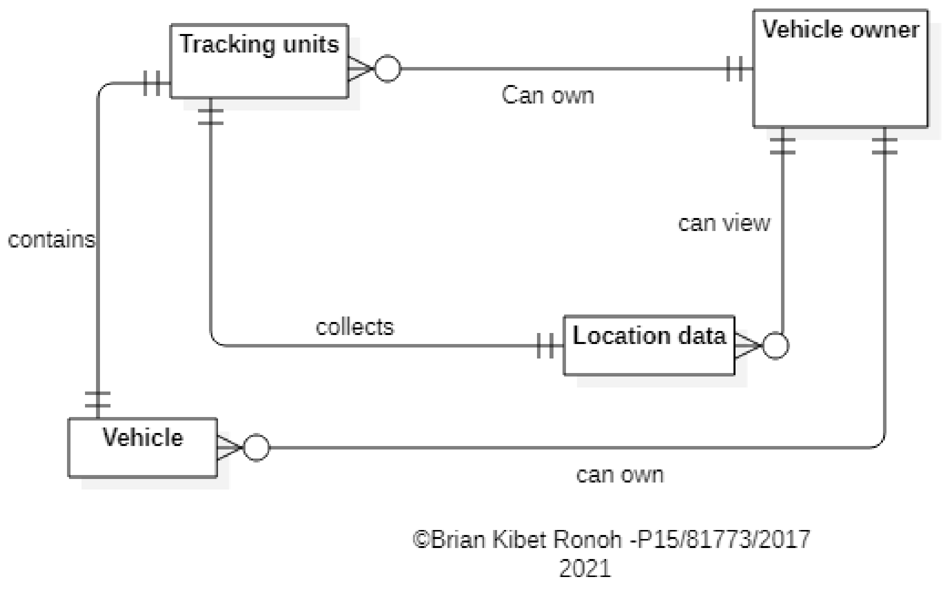


Figure 11: Entity relationship diagram

A vehicle owner can own many vehicles and thus can install a tracking unit in each vehicle. Consequently, the owner can view location data and history of the cars he/she owns. A tracking unit collects only location data for the vehicle which it is installed in.

Entities and their attributes:

Vehicle owner

- Name
- Username
- Password.
- Phone number.

Tracking unit.

- IMEI number of GSM module
- SIM IMSI number

Vehicle.

- Make
- Model
- Year
- Type
- Plate Number
- Color
- Image

Location data

- Latitude.
- Longitude.
- Current timestamp.
- Speed
- Direction

5. Use cases.

To identify, clarify and organize system requirements, use cases are used. The use case is made up of a set of possible sequences of interactions between the system and users in a particular environment and related to a particular goal.

Elements of the use case:

- **Actors** – Tracking unit and Car owner.
- **Goal** - to locate the location of a vehicle in real time.
- **The system.** The processes and steps taken to reach the end goal, including sensing location data, sending location data and showing them on a base map.

Use cases diagram.

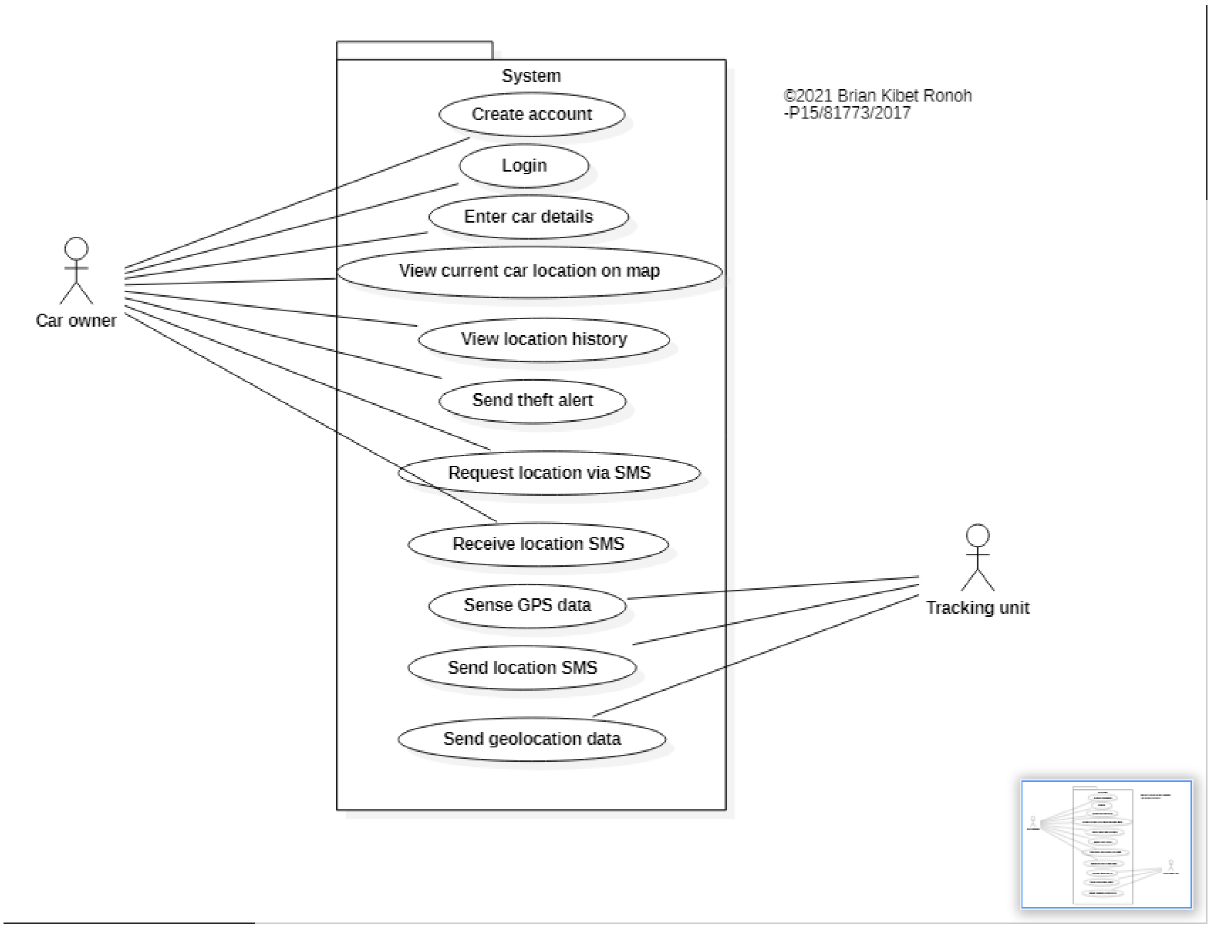


Figure 12: Use case diagram

Use case specifications

In order to use the functionality of each use case, and reflect the sequential use case dependencies, each functionality with preconditions, post conditions, actors involved in the use case and a description of the requirement is documented as follows:

Use case 1: Login

Table 1: Login use case

Use case name	**Login**
Use case ID	**001**
Description	End users (car owners) log in the system by providing their username and password.
Actors	Car owner

Precondition	System is idle, showing welcome message and a login interface
Main flow	1) The Use Case starts when the car owner selects login. 2) The car owner enters the login username and password. **3)** The car owner is logged in the system.
Post conditions	The user or administrator is logged in the system

Use case 2: Create account.

Table 2: Create account use case

Use case name	**Create Account**
Use case ID	**002**
Actors	Car owner
Description	A car owner will be able to create an account by proving required information including name(s), username, email address, password, phone number and IMEI of the GPS tracking unit.
Precondition	A valid IMEI of the GPS tracker, phone number, email address and username must be provided by the end user
Main flow	1) This use case is required when a new request of creating a new account is received. 2) The car owner starts this use case by clicking on create account on the menu. 3) The car owner provides a valid IMEI of a GPS tracker.

	4) The car owner fills the required fields of the creation form of the desired account.
Post conditions	Two tabs will be displayed. The user will have the choice to view current car location, view car state details such as speed. This will be the homepage.

Use case 3: View current car location.

Table 3: View current car location use case.

Use case name	**View current car location.**
Use case ID	**003**
Actors	Car owner
Description	The car owner views the current location of their car on a base map.
Precondition	After a successful login attempt, the system is idle, showing a base map
Main flow	1) The use case starts after the user successfully logs into the system. 2) The system requests for current location from the server. 3) Live positioning of the car is displayed.
Post conditions	Current position of the car is displayed on the map as the car moves

Use case 4: Send theft alert to local authorities.

Table 4: Send theft alert to local authorities use case

Use case name	**Send theft alert to local authorities.**
Use case ID	**004**

Actors	Car owner.
Description	The car owner reports a stolen vehicle to local authorities and optionally post in social media accounts
Precondition	The system is idle, showing an option to report that the vehicle is stolen.
Main flow	1) The use case starts when the user selects the option to report stolen car. 2) System compiles vehicle information including plate number, make, model, year of manufacture, images, last known location/current location. 3) System shows the user the drafted message 4) System asks the user for permission to send the alert to local authorities.
Post conditions	The alert has been sent and user is shown delivery status.

Use case 5: Request car location via SMS

Table 5: Request car location via SMS use case.

Use case name	**Request car location via SMS**
Use case ID	**005**
Actors	Car owner
Description	The car owner requests for the current vehicle through an SMS message.
Precondition	1) System is continuously collecting geolocation data. 2) A SIM is inserted in the GSM module of the tracking unit.
Main flow	1) The car owner texts “Location” to the IMSI number of the SIM inserted in the tracking device installed in the car.

	2) The system gets current location from the GPS module. 3) The system sends the location data back to the user containing the latitude, longitude, speed of the vehicle and a link to view the current location on the mobile app or google map via SMS.
Post conditions	The car owner successfully receives the vehicle's current location through SMS.

Use case 6: View location history.

Table 6: View location history use case.

Use case name	**View location history.**
Use case ID	**006**
Actors	Car owner
Description	The user intends to view the location history of the vehicle on a base map.
Precondition	The system is idle and showing an option to show location history for some infinite time duration i.e., 12 or 24 hours.
Main flow	1) This use case starts when the user selects the option to view location history. 2) The user selects the period to be shown where the car has been during that time. 3) The system shows the map showing the location of the vehicle during the selected period.
Post conditions	Location history is plotted on a base map that is zoomable.

Use case 7: Sense GPS data

Table 7: Sense GPS data use case.

Use case name	**Sense GPS data**
Use case ID	**007**
Actors	Arduino Uno based GPS tracking unit
Description	The GPS tracking unit senses its geolocation data using a GPS receiver and a GPS antenna.
Precondition	The GPS tracker is turned on.
Main flow	1) This use case starts when the GPS tracker is turned on. 2) After every interval of time the Arduino uno based GPS tracker will sense its geolocation data including its longitude, latitude, altitude and speed as well as other geolocation data
Post conditions	The sensed geolocation data is formatted into a readable format and sent to the web server.

Use case 8: Send Geolocation data

Table 8: Send Geolocation data use case

Use case name	**Send geolocation data**
Use case ID	**008**
Actors	Arduino Uno based GPS tracking unit
Description	The GPS tracker sends its geolocation data to the system at each interval of time
Precondition	The GPS tracker is turned on.
Main flow	After every interval of time the Arduino Uno based tracking unit sends its geolocation data

	including its longitude, latitude, altitude and speed.
Post conditions	Geolocation data will be intercepted by a web server and stored in a MySQL database

Use case 9: Send location via SMS to user

Table 9: Send location via SMS to user use case

Use case name	**Send location via SMS to user**
Use case ID	**009**
Actors	Arduino Uno based GPS tracking unit
Description	The GPS tracker sends its geolocation data via SMS to the user upon request by the user.
Precondition	1) The user has sent an SMS requesting the location of the vehicle to the number of the SIM inserted in the tracking unit. 2) The GPS tracker is turned on.
Main flow	1) The GPS tracking unit receives the SMS requesting the current location of the car. 2) The tracking unit gets the current location from the GPS antenna. 3) The tracking unit formats the received data into human readable format. 4) The tracking unit drafts message containing location information. 5) The system sends the SMS containing location details to the user via GSM network.
Post conditions	The system successfully sends the SMS to the user requesting the location of the vehicle.

CHAPTER 5: SYSTEM DESIGN.

Introduction

This chapter describes the physical designs that will meet the specifications described in the system requirements. The tasks will include user interface design, data design and system architecture.

Approach

To design the system, I used a top-down method in which I first described the high-level architecture of the system and then gradually defined the sub components of each component as I refined every module to the detailed level.

System architecture.

The first step in the designing of the vehicle tracking system is the segmentation of the system into its fundamental components. The proposed vehicle tracking system consists of three main components namely; the tracking unit, the web server and the mobile application client.

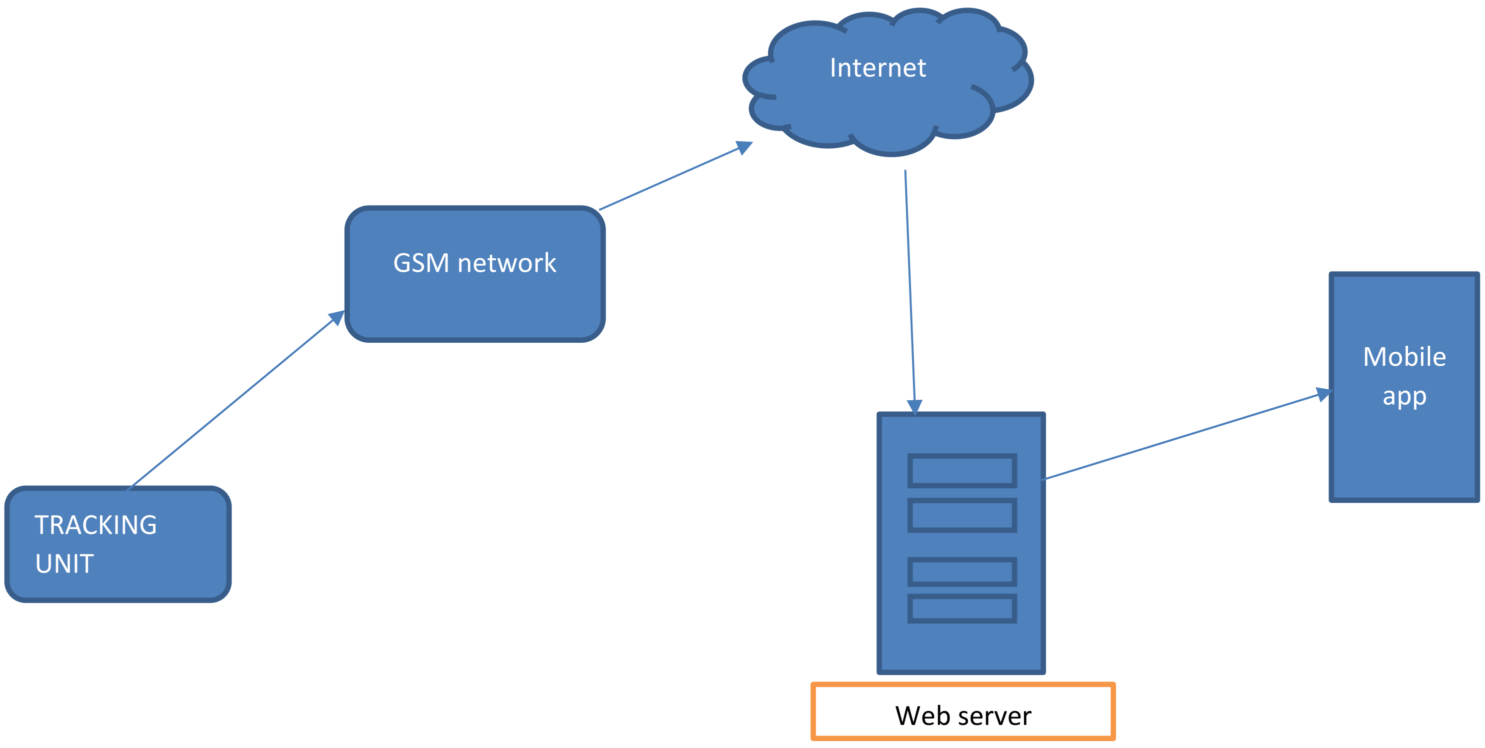


Figure 13: System architecture

The system is split into the sub-systems shown in the diagram above. The main components are discussed below:

a) **Tracking Unit** – This is an IoT device installed in the vehicle to be tracked which tracks its movement and relays this information to the server in real-time.
b) **Web server** – The tracking server has three responsibilities: receiving data from the GPS tracking unit, securely storing it, and serving this information on demand to the user. This contains the business logic and databases required to track a vehicle.
c) **Mobile app client** – this provides an interface to vehicle owners through which they can track their vehicles in real-time.

All these components are connected via a network. The IoT tracking device sends geolocation data to the web server through the internet to the web server containing a database. The mobile application client should be connected to the internet so it receives the location data. A user may request for location data from the tracking device directly through the GSM network via SMS and the system responds using the same GPS network.

Network Architecture.

The system will comprise of a mobile application. The application will utilize a 3-layered thin-client-server architecture. This architecture is mainly entails having 3 layers namely the thin-client, middle tier and backend-data server. The thin-client is the part of the application that the end user sees and does not execute much application logic but instead sends HTTPS requests to the middle-tier layer to access data or a service. The middle-tier layer performs most of the work as it mostly has the web server which receives the inbound HTTP requests and routes them to the wireless application server which in turn takes these requests and executes the appropriate logic. This layer executes services such as data access from database and data analysis. The back-end data server is where the database resides and allows services to access its data for use.

The GPS tracking unit sends data to the database in the web server by using HTTP POST requests through GPRS. The diagrammatical representation of the architecture is as shown below:

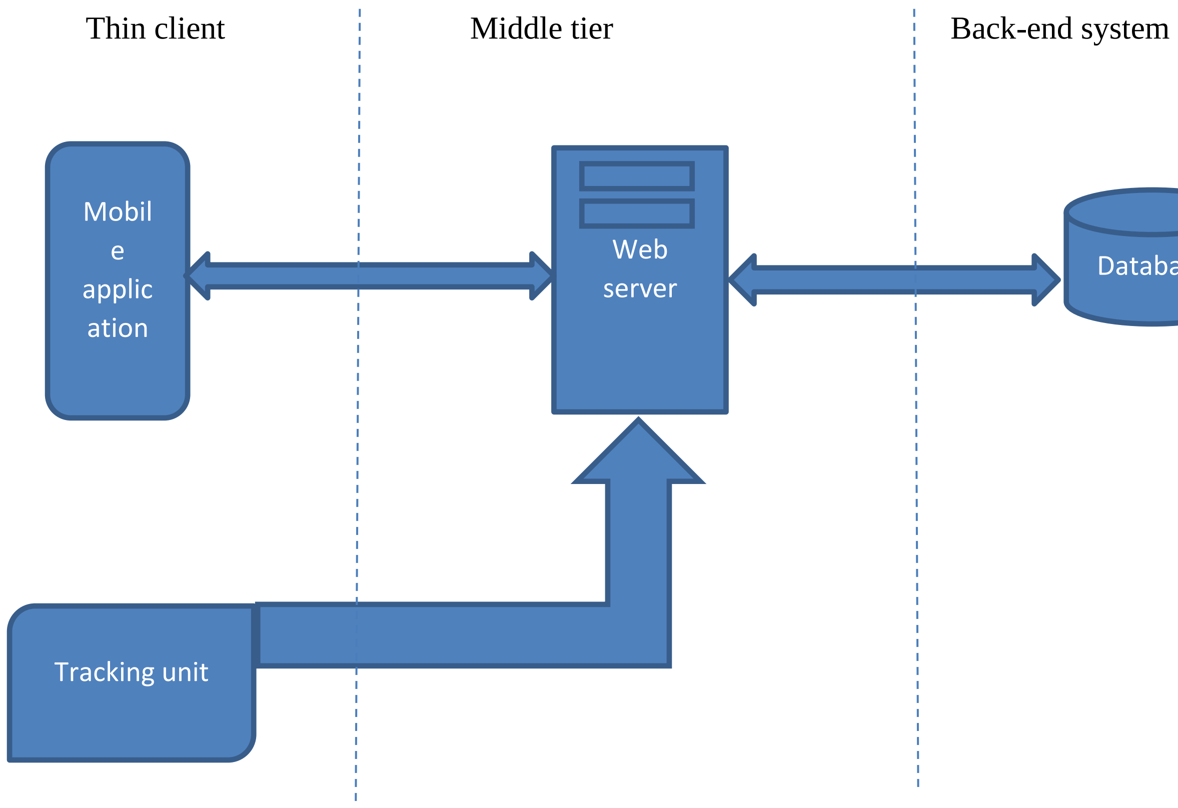


Figure 14: Network architecture of the system

Hardware design.

The tracking unit will be composed of hardware which will be installed on the vehicle. This will be the system's main source of data. The hardware used here are: SIM 808 GPS/GPRS/GSM module, Arduino Uno Microcontroller, GPS active antenna, GSM antenna and a power supply. The components will be interconnected as shown in the diagram below:

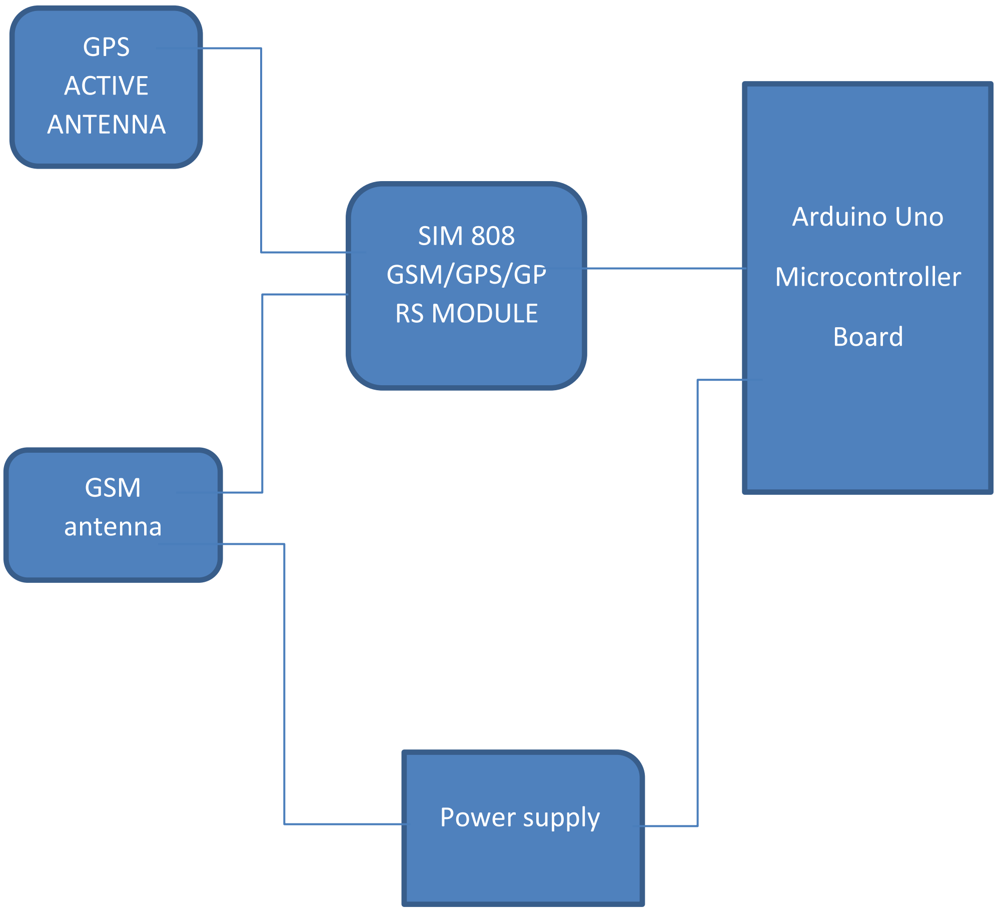


Figure 15: Tracking unit device components.

The core part of the tracking system is the Arduino Uno microcontroller. It controls the whole processing of collecting and transmitting geolocation data. The geo location of a vehicle can be captured through GPS receiver and that data will be transmitted to the web server by using GSM technology.

The SIM808 module is initialized to start gathering geo location data from the satellite; device initialization is done using AT commands and includes GPS and GSM module; to turn on the GPS, first it is powered on and put in reset mode. Then the module become ready for receiving coordinates from satellite. The GPRS is next turned on; the process includes GPRS power on, setting APN of service provider, initiating HTTP protocol, and setting protocol method (Get method). Device initialization process may take up to 1 minute to worm up and calculate the accurate position. In case of network un-availability, the acquisitioned GPS coordinates and other data such as time and speed are stored temporarily until the network returns back to service then the stored coordinates are sent with their time stamp and speed. SIM808 requires 2A peak current. So, external power supply like 12V-2A battery is used to provide the power.

GPS antenna and GSM antenna are connected to the port of SIM908 module. The SIM 808 module is connected via a serial port to the Arduino microcontroller which is loaded with code to collect geolocation data. Uploading program into Arduino is done by using Arduino IDE software.

Circuit diagram.

Circuit diagram of the GPS- and GSM-based vehicle tracking system is shown in the figure below. It is built around SIM 808 GPS/GPRS/GSM module, Arduino Uno Microcontroller, GPS active antenna, GSM antenna and a power supply

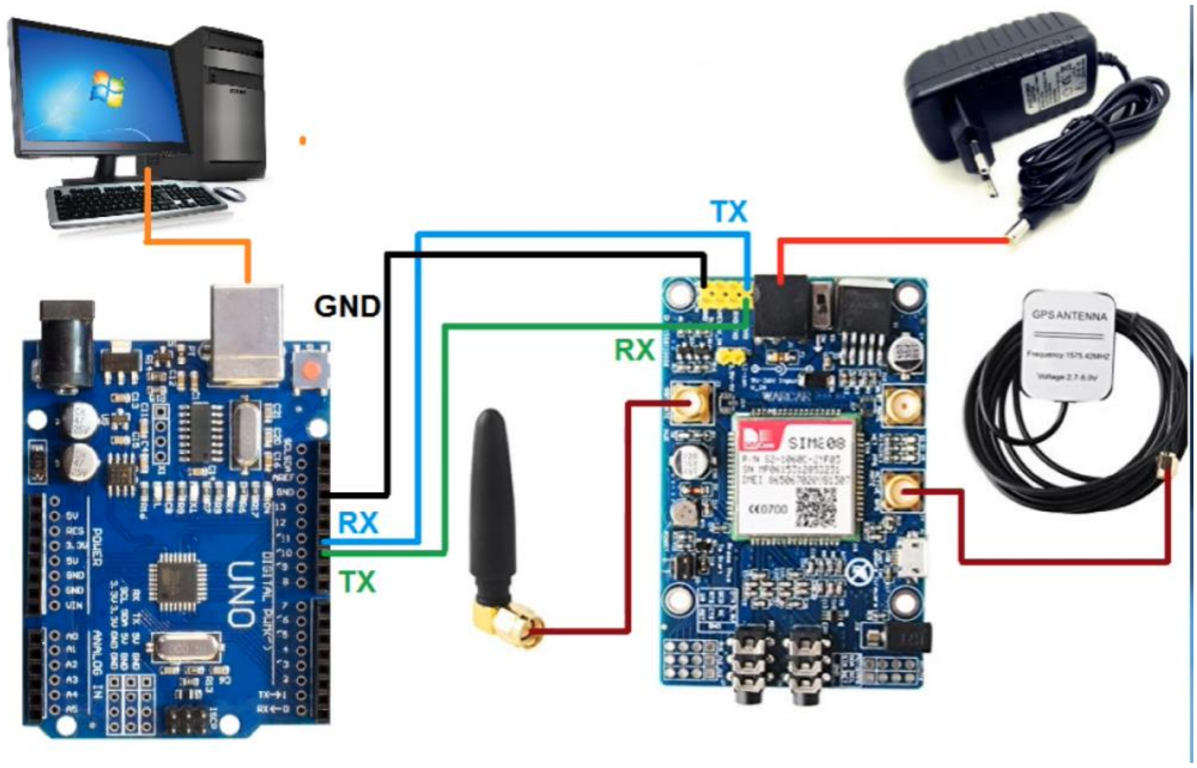


Figure 16: Circuit diagram for the tracking unit, Source: (SIM808 GPS / GSM Locator with Arduino, 2021)

The TX pin of the SIM808 module is directly connected to digital pin number 11 of Arduino Uno microcontroller and the RX pin of the SIM 808 module connected to digital pin number 10. The GND pin of the SIM 808 module is connected directly to the GND pin of Arduino Uno. The GPS and GSM antenna are connected to their respective ports in the SIM 808 module. The SIM 808 module is powered by a 9V-2A AC to DC power supply adapter while the Arduino board will be powered by the laptop/computer through the USB cable. Arduino code is loaded to the microcontroller is also loaded into the board via the USB connection.

Database Design.

The following design will help in showing how data will be stored organized and managed in tables in the database. The database for the vehicle tracking system entails the tables shown below together with the entities present within them and their properties:

Database schema

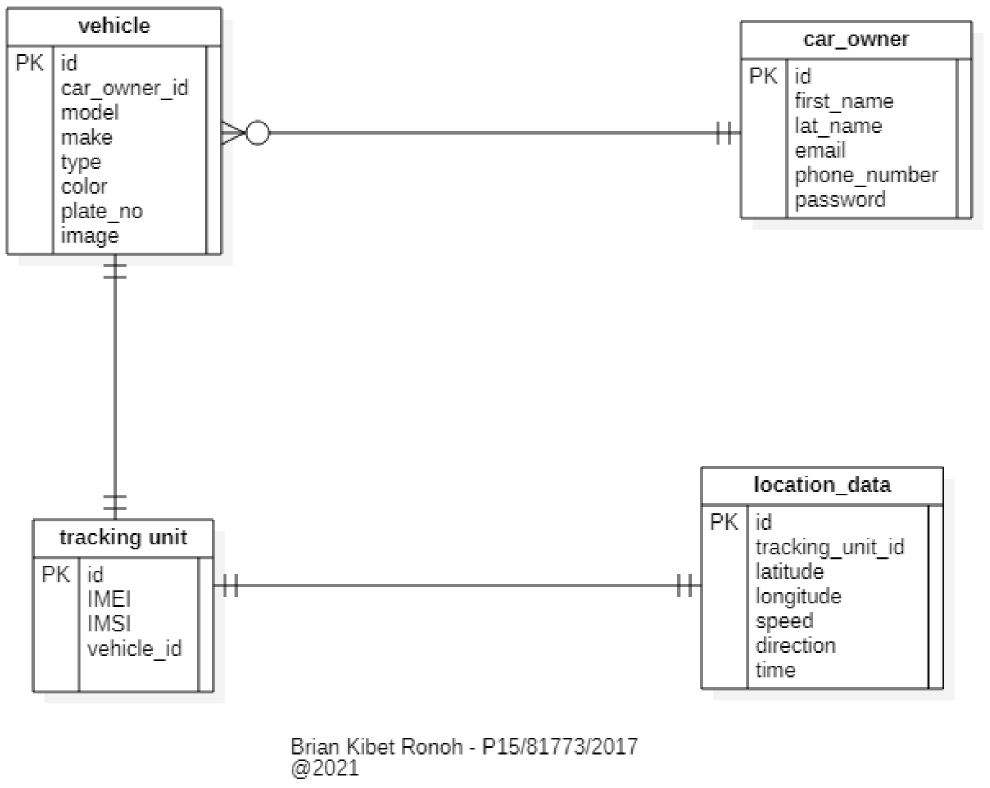
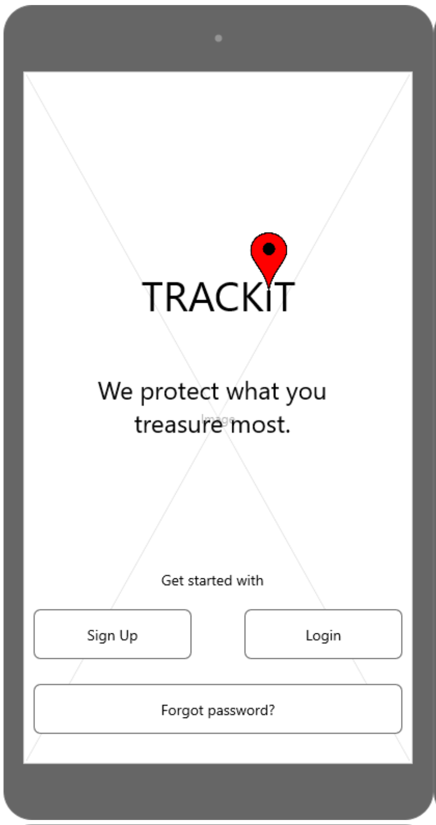
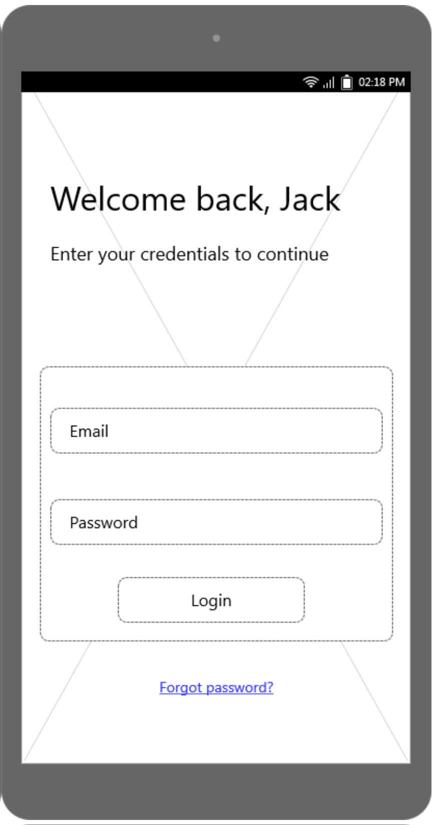
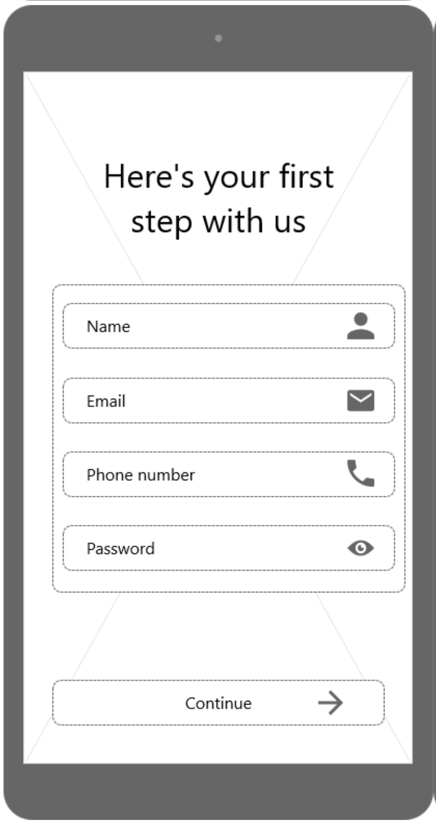


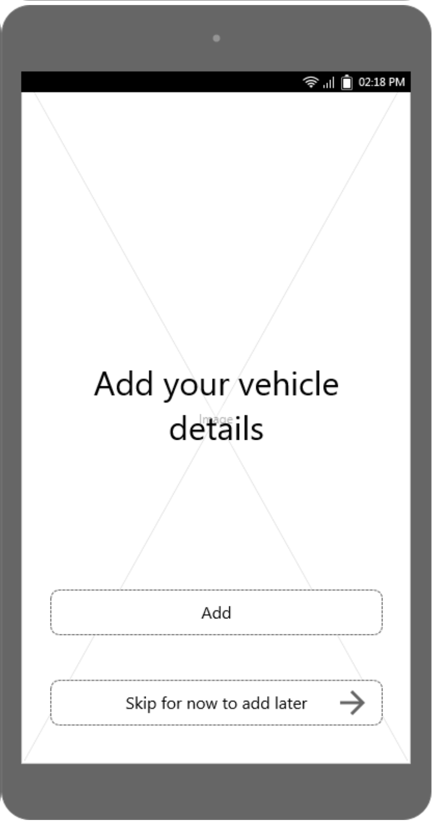
Figure 17: Database schema design.

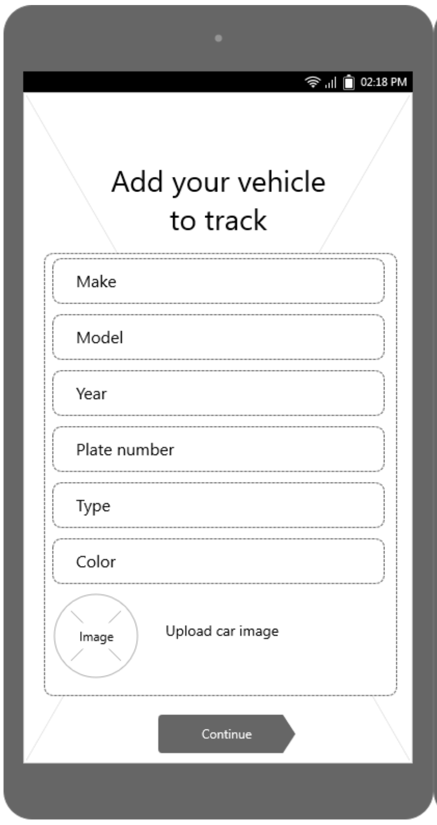
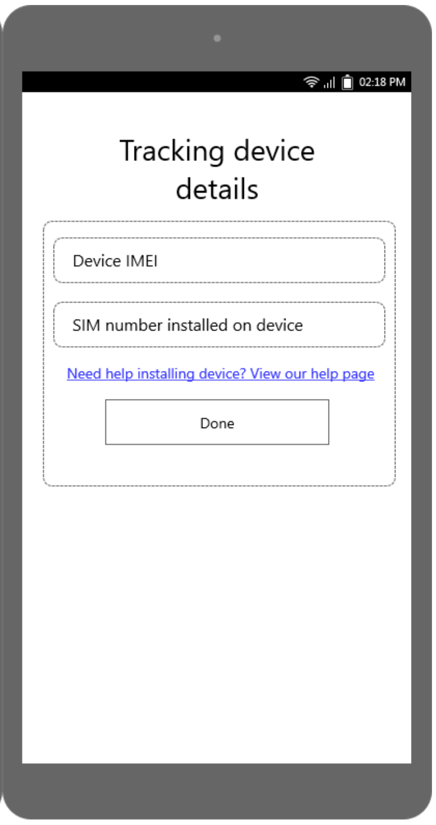
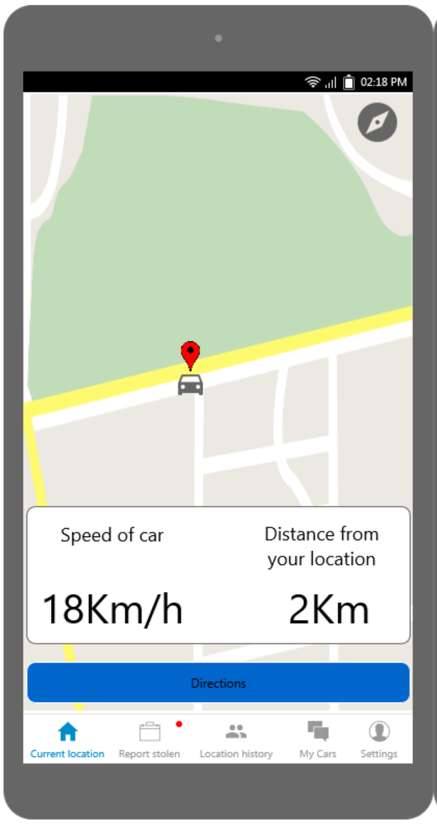
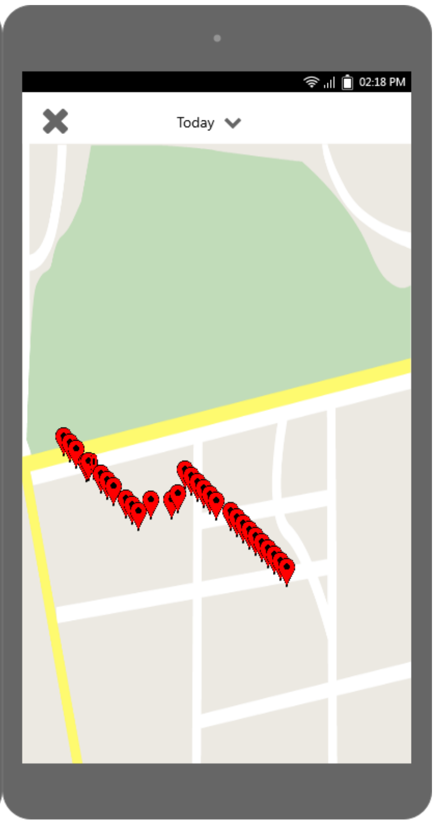
User interface design.

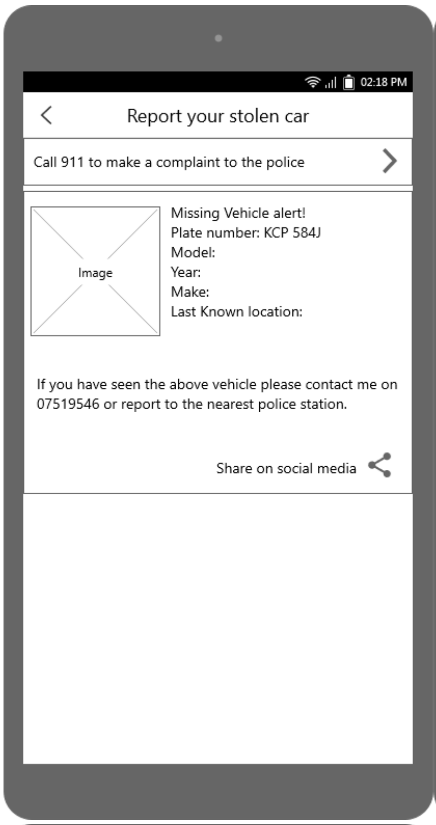






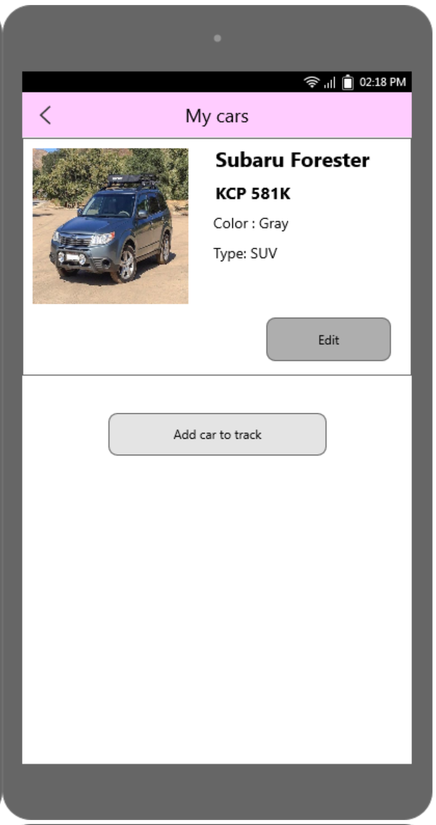


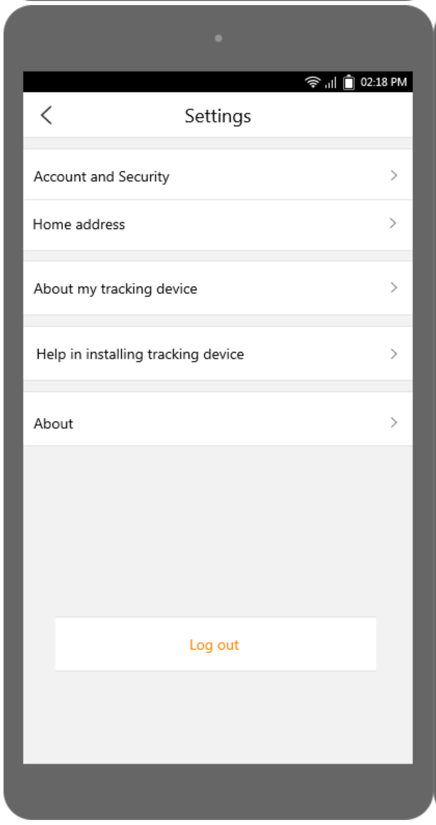

02:18 PM
Add your vehicle to track
Make
Model
Year
Plate number
Type
Color
Image
Upload car image
Continue

02:18 PM
Tracking device details
Device IMEI
SIM number installed on device
Need help installing device? View our help page
Done

02:18 PM
Speed of car
Distance from your location
18Km/h
2Km
Directions
Current location
Report stolen
Location history
My Cars
Settings

02:18 PM
Today

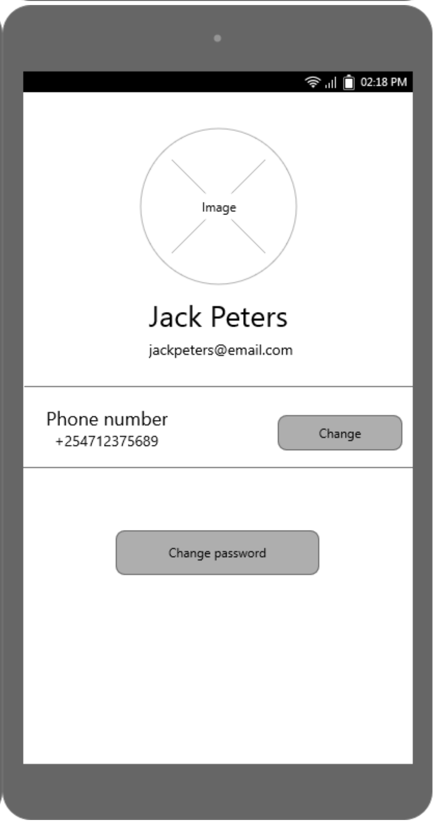


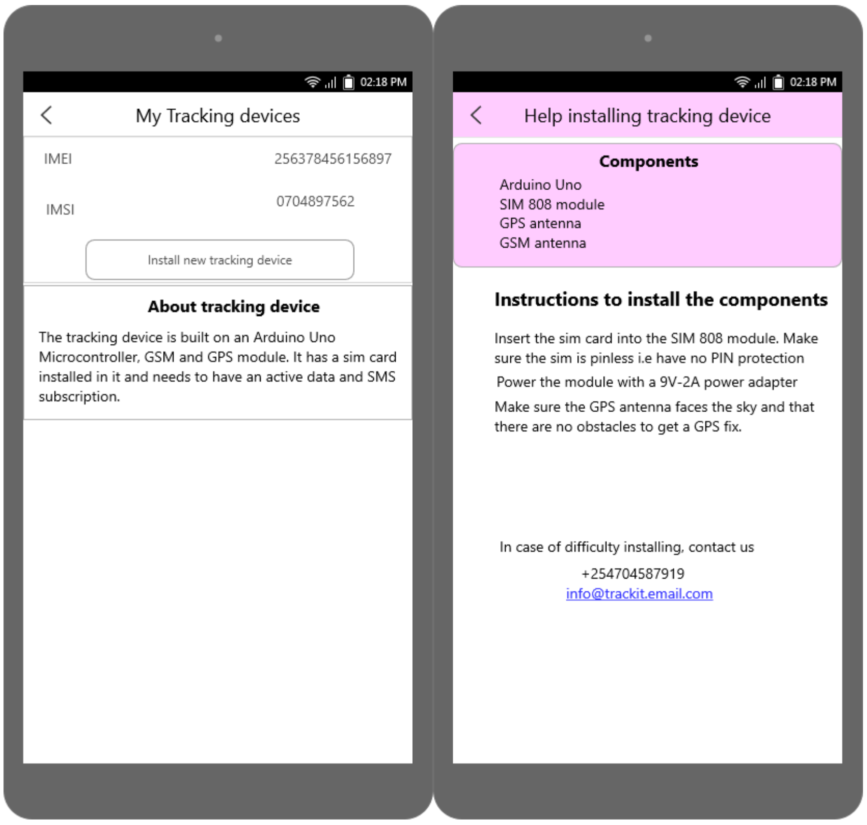
Missing Vehicle alert!
Plate number: KCP 584J
Model:
Year:
Make:
Last Known location:

If you have seen the above vehicle please contact me on 07519546 or report to the nearest police station.








02:18 PM

My Tracking devices

IMEI 256378456156897

IMSI 0704897562

Install new tracking device

About tracking device

The tracking device is built on an Arduino Uno Microcontroller, GSM and GPS module. It has a sim card installed in it and needs to have an active data and SMS subscription.

02:18 PM

Help installing tracking device

Components

Arduino Uno
SIM 808 module
GPS antenna
GSM antenna

Instructions to install the components

Insert the sim card into the SIM 808 module. Make sure the sim is pinless i.e have no PIN protection

Power the module with a 9V-2A power adapter

Make sure the GPS antenna faces the sky and that there are no obstacles to get a GPS fix.

In case of difficulty installing, contact us

+254704587919
info@trackit.email.com

Figure 18: User interface Android pages designs

CHAPTER 6: SYSTEM IMPLEMENTATION AND TESTING

Introduction.

This chapter contains the details of my findings during the implementation and testing stage. It will also contain the hardware and software resources that were used to realize the implementation of the modules together with the programming languages.

Technology enablers.

The technology enablers are defined as a set of features and technology that will be used in the implementation of the application. The implementation of the system consists of the hardware and the software part. The following technologies were used in the implementation of the system.

IoT hardware.

The following IoT components were used to implement the system hardware:

i. Arduino Uno
ii. SIM 808 GPS/GSM module.
iii. GSM antenna.
iv. GPS antenna.
v. 9V 2A power adapter.

Programming tools and technologies.

Java.

Java programming language alongside XML was used to implement the Android app. The language was chosen because it is the official language for Android development and it is the most supported language by Google.

C++

Embedded C++ was used to implement the IoT software to send the coordinates to the server, receive incoming SMS location request and send the location data to user via SMS.

PHP

PHP is a server-side scripting language that was designed for web development. However, it is also used as a general-purpose programming language. In this project, it is used to read POST HTTP requests sent by the Arduino board, and store the GPS data in a MySQL database. It is also to generate a JSON file containing the GPS data related to the current position of a car.

XML

XML or Extensible Markup Language is a markup language that defines a set of rules to encode documents. In this case, it was used to design UI layouts and screen elements constituting our Android application.

JSON

JSON or JavaScript Object Notation, is an open standard format that has a simple readable structure, and that serves in transmitting data between a server and web application. In this project JSON file was generated using a PHP program and contained GPS data such as longitude, latitude, altitude and speed.

Android studio.

Android Studio IDE was used to develop the mobile app. Android Studio is the official integrated development environment for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. Android Studio provides the fastest tools for building apps on every type of Android device.

Arduino IDE.

The Arduino IDE was used to write embedded C++ code and upload it to the Arduino Uno. The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.

MySQL database.

MySQL is the most widely used open-source relational database management system, and uses the Structured Query Language (SQL). In this project, MySQL is used to store information about the car including GPS coordinates, moving speed and followed itinerary.

Google Maps Android SDK.

The Google Maps SDK was used to integrate Google Maps to the Android App to show the location of the vehicle. With the Maps SDK for Android, developers can add maps to their Android apps including Wear OS apps using Google Maps data, map displays, and map gesture responses. Developers can also provide additional information for map locations and support user interaction by adding markers, polygons, and overlays to your map.

The SDK supports both the Kotlin and Java programming languages and provides additional libraries and extensions for advanced features and programming techniques. (Google Developers, 2021)

<u>Git.</u>

Git was heavily incorporated during the product development, where a repository was created in GitHub and all the changes were pushed for both backup and management purposes. The project can be found in the following URL: https://github.com/kibet-brian-ronoh/TrackIt

Implementation results.

The implementation process involved five major phases:

i. IoT hardware assembling and implementation.
ii. IoT software installation and programming.
iii. Database server set up.
iv. IoT sending location data to database.
v. Android application development.

These three modules of the development process birthed one unified system that was seeking to fulfill the objectives set. In each phase, testing was done to ensure the system worked in the desired way.

<u>i. **IoT hardware assembling and implementation**</u>

<u>Hardware installation.</u>

The figure below shows the complete hardware installation of the IoT device.

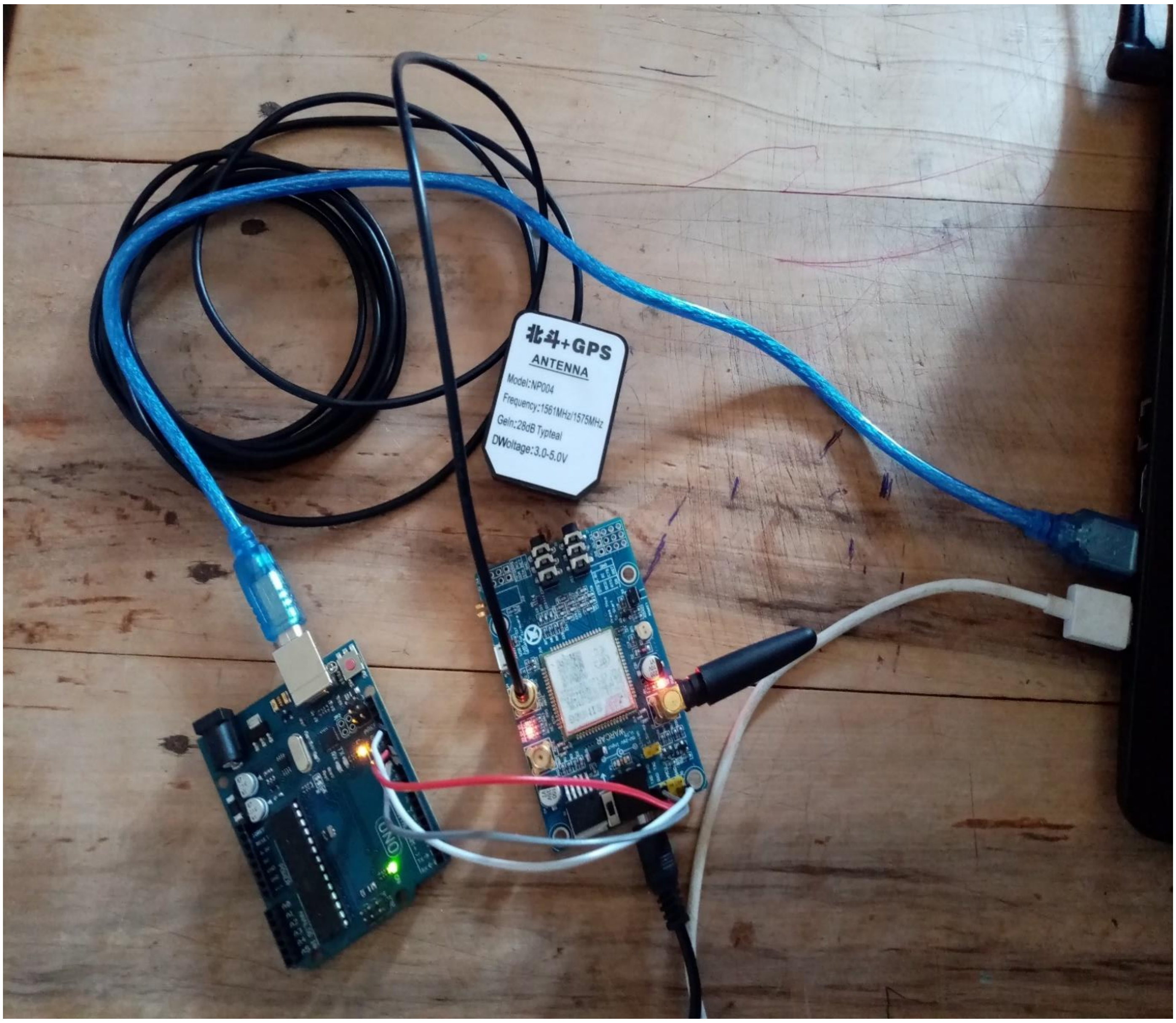


Figure 19: Complete hardware connection of the IoT device.

The SIM card was unlocked and inserted on to the SIM card holder of the SIM 808 module, the GSM and GPS antennas were connected, and the power adapter connected to the SIM808 module. The 9V battery was left for the final installation as the USB cable used for programming and the power adapter provided power supply needed for the initial set up and testing.

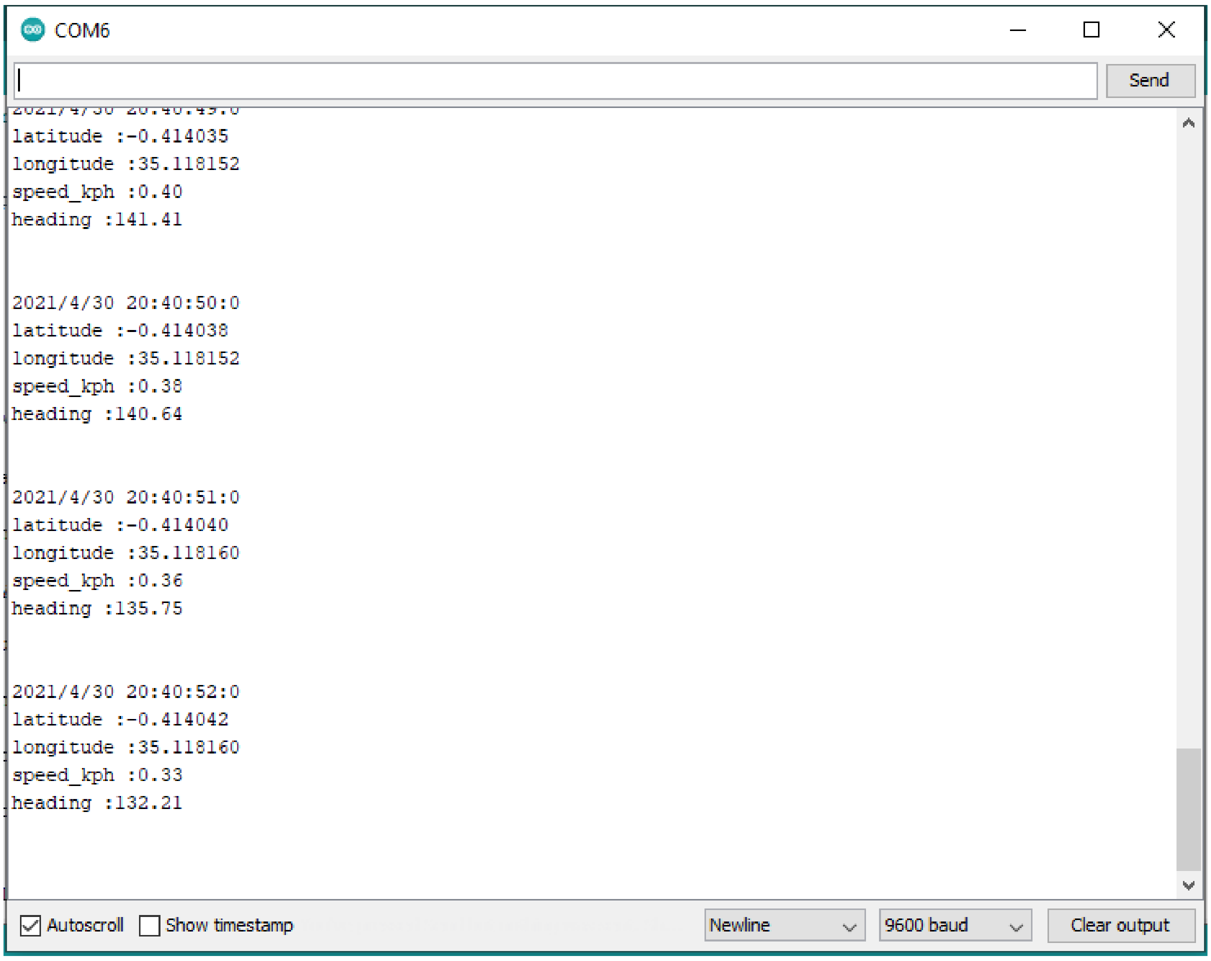
ii. IoT Software installation and programming.

Once the hardware installation had been completed, the project moved to the next implementation phase, i.e., programming of the Arduino to track the speed and location. Arduino IDE was needed for programming. The two available options were to download the Arduino IDE or use the Arduino Web Editor. The former was chosen as it provided an easy setup with just an installation of the Arduino IDE being required.

Next, the program was written. There exists a wide number of libraries from the Arduino community. Tiny GPS, Tiny GPS++, DF Robot, SIM808 and Adafruit Fona are most of the libraries that were studied and experimented with. The project finally settled on DF Robot library as a personal preference as it turned out to be more stable and simplified.

The main output of this phase of implementation was to get the speed and the coordinates i.e., latitude and longitude from the IoT device. This phase also included sending the location data via SMS to the user on request. The user texts the number in the IoT device, the system receives the text message, extracts the incoming message sender's number and replies back with the location details including a link to view the map in Google Maps.

Testing of the device was achieved through printing the coordinates received from the GPS receiver on the serial monitor of the Arduino IDE. This was achieved by connecting the Arduino board to the computer with a USB cable and switching the board ON with the ON/OFF button. The baud rate on the monitor was selected to match the baud rate in the code i.e., the recommended 9600 bps. The code was then compiled and after fixing of the errors, it was uploaded to the board. The output of the location co-ordinates and speed from the Arduino IDE serial monitor is shown below:

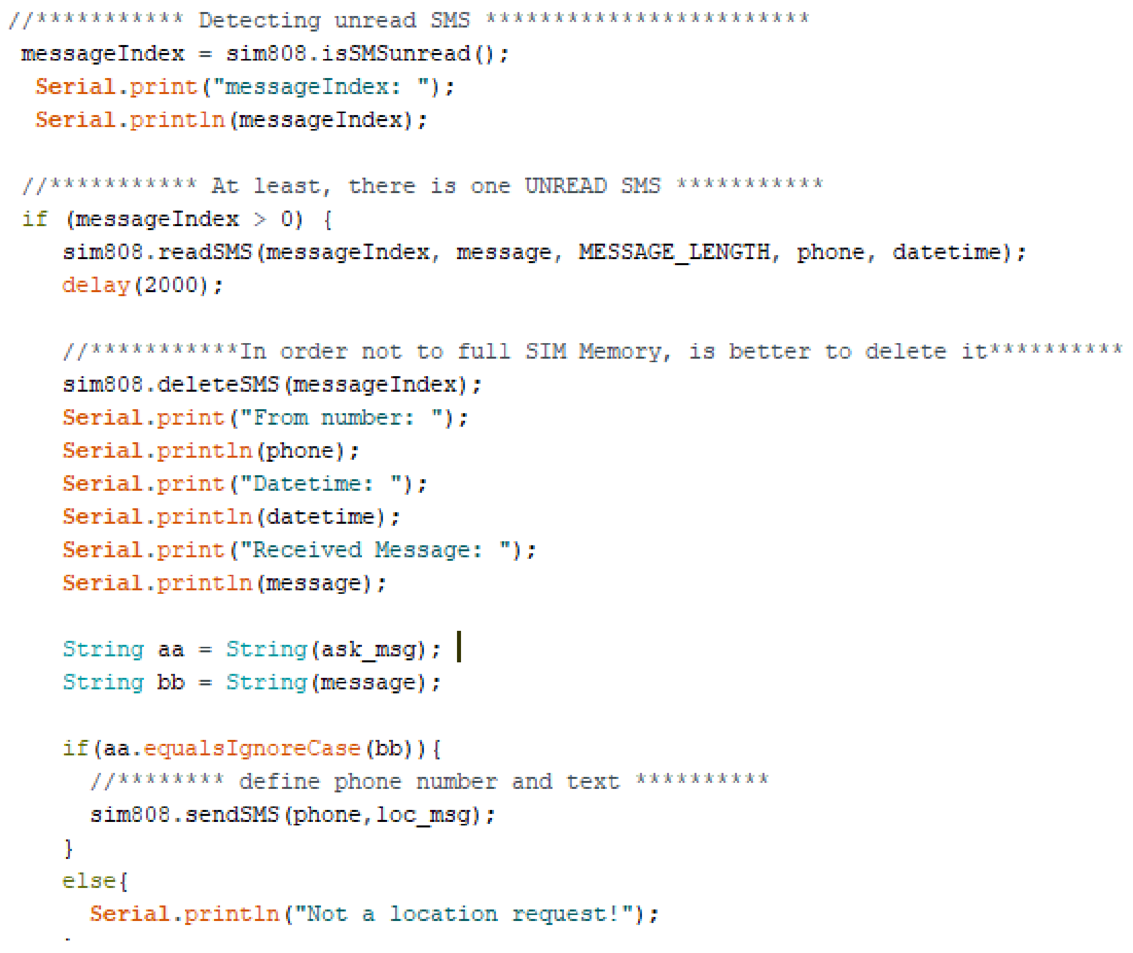


```
latitude :-0.414035
longitude :35.118152
speed_kph :0.40
heading :141.41


2021/4/30 20:40:50:0
latitude :-0.414038
longitude :35.118152
speed_kph :0.38
heading :140.64


2021/4/30 20:40:51:0
latitude :-0.414040
longitude :35.118160
speed_kph :0.36
heading :135.75


2021/4/30 20:40:52:0
latitude :-0.414042
longitude :35.118160
speed_kph :0.33
heading :132.21
```

Figure 20: The output of the location co-ordinates and speed on the serial monitor

A new location is obtained every second as the GPS module sends this data to Arduino.

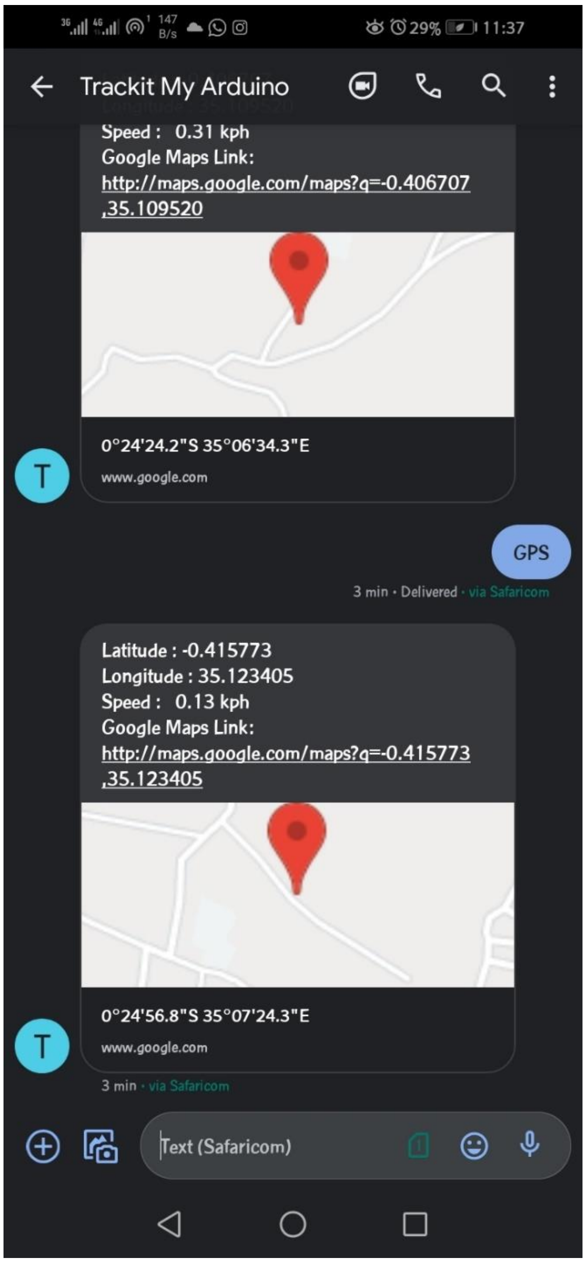
For sending an SMS of the location to user, the code highlight below was used:



```
//*********** Detecting unread SMS ************************
 messageIndex = sim808.isSMSunread();
  Serial.print("messageIndex: ");
  Serial.println(messageIndex);

 //*********** At least, there is one UNREAD SMS ***********
 if (messageIndex > 0) {
    sim808.readSMS(messageIndex, message, MESSAGE_LENGTH, phone, datetime);
    delay(2000);

    //***********In order not to full SIM Memory, is better to delete it**********
    sim808.deleteSMS(messageIndex);
    Serial.print("From number: ");
    Serial.println(phone);
    Serial.print("Datetime: ");
    Serial.println(datetime);
    Serial.print("Received Message: ");
    Serial.println(message);

    String aa = String(ask_msg);
    String bb = String(message);

    if(aa.equalsIgnoreCase(bb)){
      //******** define phone number and text **********
      sim808.sendSMS(phone,loc_msg);
    }
    else{
      Serial.println("Not a location request!");
```

Figure 21: Code highlight for sending Location SMS to user

The screenshot from a mobile device shows the result of the implementation of the sending location data via SMS feature.


11:37

Trackit My Arduino

Speed : 0.31 kph
Google Maps Link:
http://maps.google.com/maps?q=-0.406707,35.109520

0°24'24.2"S 35°06'34.3"E
www.google.com

GPS

3 min · Delivered · via Safaricom

Latitude : -0.415773
Longitude : 35.123405
Speed : 0.13 kph
Google Maps Link:
http://maps.google.com/maps?q=-0.415773,35.123405

0°24'56.8"S 35°07'24.3"E
www.google.com

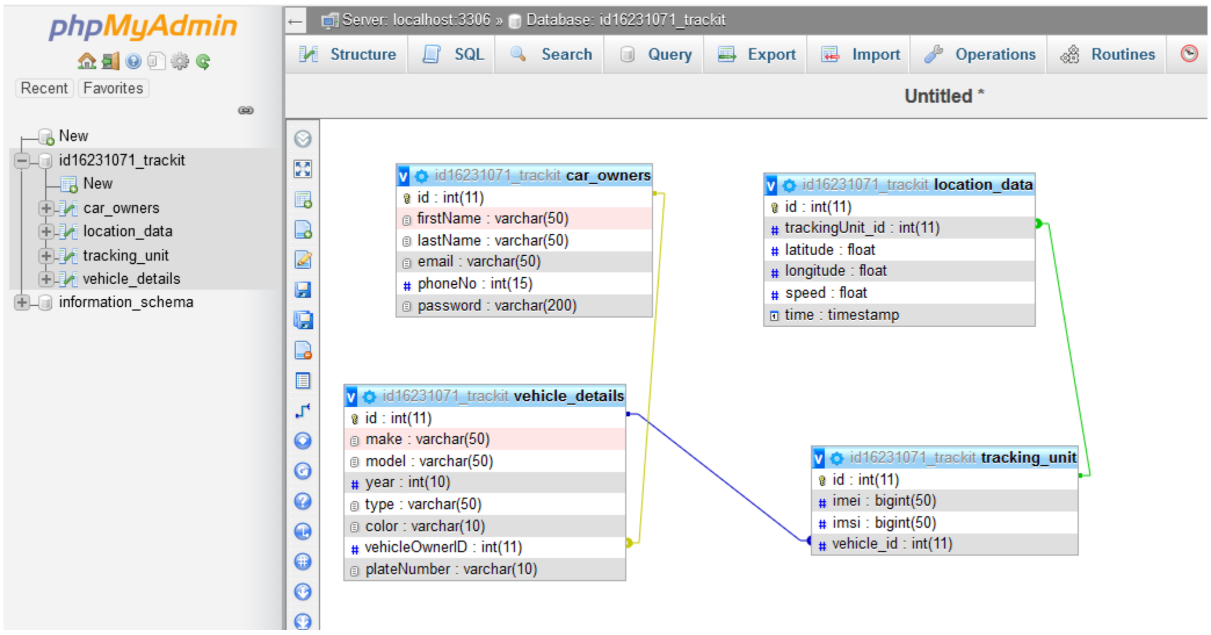
3 min · via Safaricom

Figure 22: Screenshot of mobile requesting for location

iii. **Database server setup.**

The implementation of this cycle was to setup the Server, which in this case was the MySQL database so that it could receive GPS data from the Arduino tracking device. The database was also designed to store the user data e.g., vehicle details, personal information, tracking device details and location data.

The database was then uploaded to Hostinger Web Host (000webhost.com) for hosting as the IoT device could not send location data to a local machine through GPRS. The below screenshot of the phpMyAdmin console shows the schema implementation:



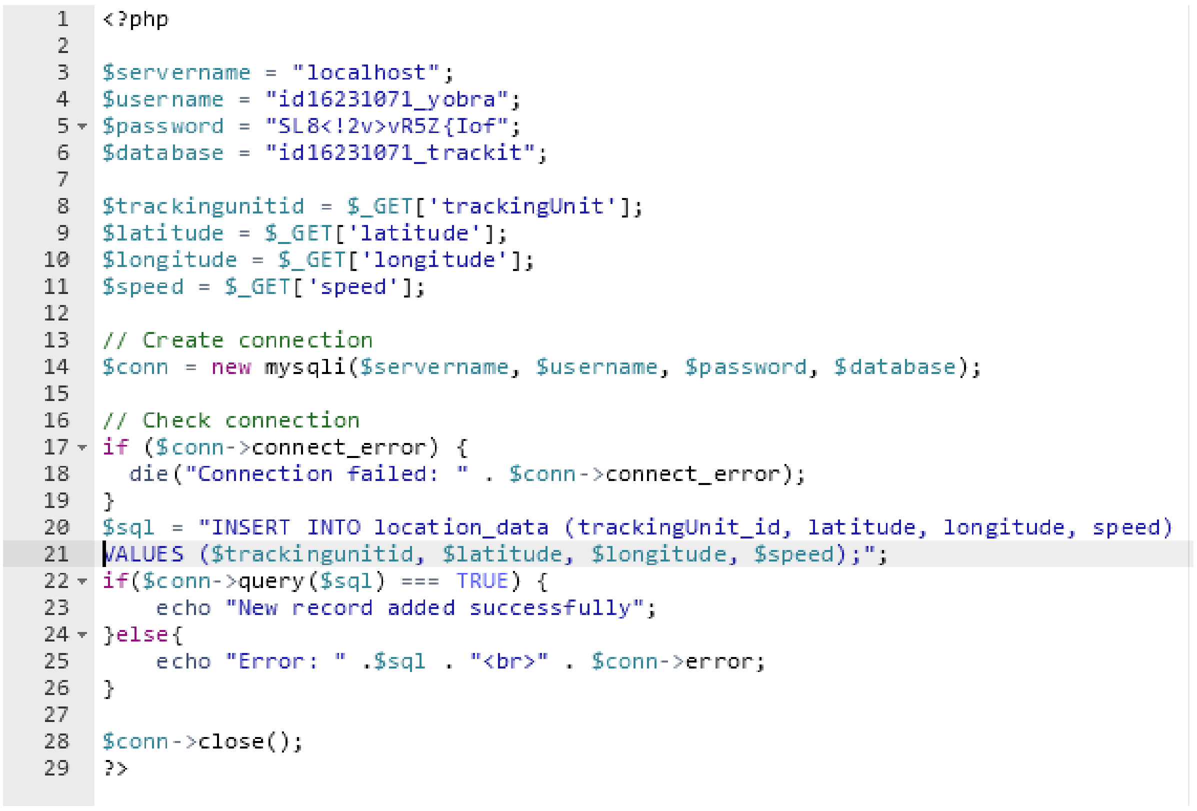
```
id16231071_trackit car_owners
id : int(11)
firstName : varchar(50)
lastName : varchar(50)
email : varchar(50)
phoneNo : int(15)
password : varchar(200)

id16231071_trackit location_data
id : int(11)
trackingUnit_id : int(11)
latitude : float
longitude : float
speed : float
time : timestamp

id16231071_trackit vehicle_details
id : int(11)
make : varchar(50)
model : varchar(50)
year : int(10)
type : varchar(50)
color : varchar(10)
vehicleOwnerID : int(11)
plateNumber : varchar(10)

id16231071_trackit tracking_unit
id : int(11)
imei : bigint(50)
imsi : bigint(50)
vehicle_id : int(11)
```

Figure 23: Database schema implementation

Writing php scripts (REST API)

After successful database set up, php API scripts were written to store and retrieve data from the database. The code highlighted below (storeLocation.php) creates an interface that links the database server to the Arduino. It allows the IoT device insert the location data in the database for retrieval from the Android app.

/public_html/storeLocation.php



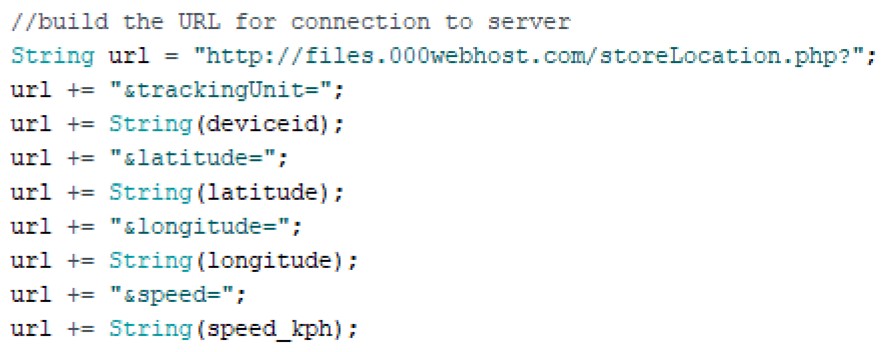
```php
<?php

$servername = "localhost";
$username = "id16231071_yobra";
$password = "SL8<!2v>vR5Z{Iof";
$database = "id16231071_trackit";

$trackingunitid = $_GET['trackingUnit'];
$latitude = $_GET['latitude'];
$longitude = $_GET['longitude'];
$speed = $_GET['speed'];

// Create connection
$conn = new mysqli($servername, $username, $password, $database);

// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}
$sql = "INSERT INTO location_data (trackingUnit_id, latitude, longitude, speed)
VALUES ($trackingunitid, $latitude, $longitude, $speed);";
if($conn->query($sql) === TRUE) {
    echo "New record added successfully";
}else{
    echo "Error: " .$sql . "<br>" . $conn->error;
}

$conn->close();
?>
```

This PHP program serves to connect to the MySQL database using the server address, database name, database username and database password. Whenever the GPS tracker sends HTTP requests, this program will intercept these requests and store the received variables into local variables.

After reading the GPS variables and storing into local variables, we attempt to connect to MySQL database, and on success, we store the received variables in their corresponding table columns.

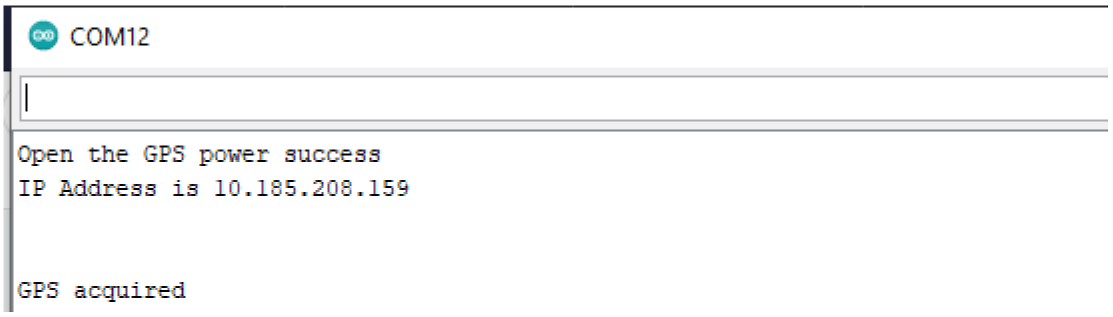
iv. <u>IoT sending location data to database.</u>

This phase of implementation dealt with the communication between the Arduino and the web server containing the database. This was achieved with additional programming of the Arduino using the DFRobot library code as seen in the Appendix section.

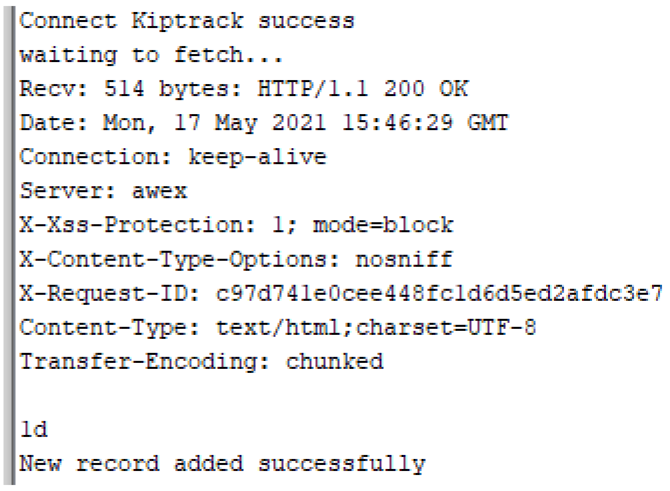


```
//build the URL for connection to server
String url = "http://files.000webhost.com/storeLocation.php?";
url += "&trackingUnit=";
url += String(deviceid);
url += "&latitude=";
url += String(latitude);
url += "&longitude=";
url += String(longitude);
url += "&speed=";
url += String(speed_kph);
```

The above illustrates how the URL link was built. Device id which is the tracking unit ID was appended to the base URL together with the GPS data. This was then used by the DFRobot library to implement AT commands earlier introduced, to establish an HTTP connection between the tracking device and the proxy server and send the GET request. The below is the output printed on the serial monitor on a successful HTTP POST:



```
Open the GPS power success
IP Address is 10.185.208.159


GPS acquired
```



```
Connect Kiptrack success
waiting to fetch...
Recv: 514 bytes: HTTP/1.1 200 OK
Date: Mon, 17 May 2021 15:46:29 GMT
Connection: keep-alive
Server: awex
X-Xss-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Request-ID: c97d741e0cee448fc1d6d5ed2afdc3e7
Content-Type: text/html;charset=UTF-8
Transfer-Encoding: chunked

1d
New record added successfully
```

v. Android application development.

The focus here was to design and build the Android application and connect it to the MySQL database and display the location on Google map. Android Studio, earlier introduced, was used as the development environment for the Android application.

Taking into consideration the tracking information stored in the database and what was expected to be displayed to the user, the application settled on the following main activities:

i. Login Activity.
ii. Register Activity.
iii. Car Details activity
iv. Map Activity
v. Location History Activity
vi. Tracking Device Details
vii. Help Page activity

These activities outputs are shown in the user manual which guides the user on navigating the app.

TESTING.

Software testing is a critical element of software quality assurance and represents the ultimate service of specification design and coding. Testing was based on identification and correction of errors and bugs in the system. Testing was done incrementally starting from the most basic component of the system until the system was tested as a single unit. System testing was performed to evaluate the system's compliance against the specified requirements. The functionalities of the system were tested from an end-to-end perspective.

The system was tested in three phases:

1. Unit Testing
2. Integration Testing
3. System Testing

Unit testing

This testing involved testing the system modules/components one by one so as to confirm that they perform their primary function. The components and their units tested included:

1) IoT component (Arduino and SIM 808)
2) Database component (MySQL, REST API scripts)
3) Android App. (Location Map, Location history, Vehicle Details, User authentication, Tracking Device details)

Integration testing

In this stage modules that were integrated with other modules and then tested for compatibility to check that the different modules of the system work together and to discover the faults that may arise from interaction between integrated units. The Big bang approach of integration testing was used where once all modules were successfully unit tested, they were subsequently combined together and tested one at a go. The tracking device module was integrated with the database through the API and was tested for correctness and its ability to send the location data to the server. The integrated modules worked well and the server was then integrated with the Android app via the APIs and tested.

System testing.

This involves testing the system as a whole unit so as to demonstrate that the system performs both functionally and operationally as specified.

So far, all the tests had been done with the IoT tracking device connected to the computer and with random generated data. The final phase, therefore, was to confirm these results with a practical test of the IoT device.

Unlike the previous tests, this time the Arduino tracking device was disconnected from the computer and powered with the 9V lithium battery. The pre-requisites were the sim card in the tracking device had to have some active data, and the mobile phone requesting location needed to have access to the internet.

This testing involved the following tests:

1. Functional testing
2. User acceptance testing.

Functional testing

The main aim of this test is to test the functionalities of the system. This testing usually involves use of Black-Box test-case Technique that enables one to derive a set of input conditions that will fully exercise all the functional requirements for a program. This technique is very useful as it helps discover interface errors and errors in accessing the database and performance errors. This process is aided by the test cases created in the earlier chapter of Design. The results of the testing are as follows:

Table 10: Test cases for functional testing

Test case ID	Modules	Sub-modules	Inputs	User Scenario	Expected output	Actual output
1	System authentication	Login	Email Password	Correct email and password	Map homepage is displayed	Map homepage was displayed
				Incorrect format of email or password	Error message: Invalid input	Error message: Invalid input
				Unidentified email.	Error message: Invalid email.	Error message: Invalid email.

				Wrong password	Error Message: Wrong password!	Error Message: Wrong password!
				Correct user quits	User goes to back page	User goes to back page
				No input	Error Message: Invalid input.	Error Message: Invalid input.
		Sign up	First name Last name Email Phone number Password Password confirmation	New Correct Details without duplicate email.	Account created and Add vehicle details page shown	Account was created and add vehicle details page was shown
				New Correct Details with duplicate email	Error Message: This email is already registered	Error Message: This email is already registered
				Incorrect format of Sign-up details or no input	Error Message: Invalid input	Error Message: Invalid input
		Update user account profile	Phone number or Email or Password	Correct Profile Details	Profile is updated and Alert: Profile changes made successfully.	Profile was updated and Alert: Profile changes made successfully was shown
				User Quits after viewing My Account activity	User goes to More page	User goes to More page
				Incorrect format of profile details	Error Message: Invalid input	Error Message: Invalid input
2.	Data input (Vehicle details, tracking device details)	Add vehicle details	Vehicle make, model, type, year, number plate, color	Valid vehicle details without duplicate vehicle registration number.	Details saved and user redirected to Add tracking device details page on click of button.	Details were saved and user was redirected to Add tracking device details page on click of button.
				New correct details with duplicate vehicle registration number	Error Message: Vehicle is already registered	Error Message: Vehicle is already registered
				Incorrect format of Vehicle details	Error Message: Invalid input	Error Message: Invalid input

				User with valid vehicle details Quits	User goes back to Sign up page	User goes back to Sign up page
		Add tracking device details	SIM 808 IMEI and inserted SIM card IMSI	Valid tracking device details with no duplicates.	Details saved and user is shown a successful registration pop up	Details were saved and user was shown a successful registration pop up
				Valid tracking device details with duplicates	Error Message: This tracking device is already being used on another vehicle.	Error Message: This tracking device is already being used on another vehicle
				Invalid tracking device details	Error Message: Invalid inputs	Error Message: Invalid inputs
				No input	Error Message: IMEI and IMSI cannot be empty	Error Message: IMEI and IMSI cannot be empty
3.	Report stolen vehicle	Call police hotline (911)	Button click	Click of call 911 button	The system opens the call dialer with 911 already dialed.	Dialer was opened and 911 was already dialed ready for calling
				No user click	System remains in the Report stolen vehicle page	System remained in the Report stolen vehicle page
		Share Lost vehicle alert on social media.	Button click	User clicks the share button	System shows social media icons of apps that can share the message with a ‘share via’ title	System shows social media icons of apps that can share the message with a ‘share via’ title
				No user click	System remains in the Report stolen vehicle page	System remained in the Report stolen vehicle page
4.	Show Location History	Select Date to show locations on that date.	Date in the format YYYY-MM-DD (Date Picker is also provided)	User clicks on calendar icon	Datepicker dialog is shown and the user may select past dates to view location history. The edit text field is field automatically after user selects date from the dialog.	Datepicker dialog was shown and the user was able to select past dates to view location history. The edit text field was filled automatically.
				User inputs a past date manually in a valid format.	Input accepted and user can proceed.	Input was accepted.
				User inputs a future date in a valid format.	Error message: Cannot show location history of a future date	Error message: Cannot show location history of a future date

				Invalid date format	Error Message: Invalid format of date	Error Message: Invalid format of date
				No date input	Error message: Date cannot be empty.	Error message: Date cannot be empty.
		Show map with location pins of the date selected	Previously selected date.	There is location data for the selected period.	System shows a map with location pins showing location at different times. When user clicks a location pin, the time and speed of vehicle at that location pops up.	System showed a map with location pins showing location at different times. Time and speed was displayed as location pin title.
				There is no location data for selected period	Alert dialog pops up with error: No location data for this date.	Alert dialog pops up with error: No location data for this date.
5.	Show current location	Show map with current location.	Tracking Device ID.	Tracking device is installed and running.	The most recent location itinerary i.e latitude, longitude and speed is fetched from the database and location plotted on the map.	Current location shown by the pin on the map.
				No tracking device installed or tracking device doesn't send data to web server.	Map is shown but with no location pin.	Map shown and error: No location data for this tracking device.
6.	Send location SMS to user.	Send current location details with link to google maps via SMS	Send SMS to SIM installed on tracking device.	SIM inside tracking device has an active text subscription and the user sends it a text message "GPS".	User receives message with current latitude, longitude, speed and a google maps link.	Received an SMS with current latitude, longitude, speed and Google Maps link.
				SIM inside tracking device has no active text subscription and the user sends it a text message "GPS".	User does not receive an SMS text back.	User does not receive an SMS text back.
				User sends an invalid message without the word "GPS"	User does not receive an SMS text back.	User does not receive an SMS text back.

User acceptance testing.

Due to privacy concerns, it was not possible to test parts of the system functionality by placing the hardware device on a real vehicle for live testing. However, the decision was made to test the device by walking around with the device in a bag.

The Android app was passed to about 20 motor vehicle owners and most of them gave positive feedback hence confirming that it was accepted by the clients and ready for operational use which was the main target of the testing.

CHAPTER 7: CONCLUSION.

Achievements.

The successful completion of this project resulted in the development of a vehicle tracking system that is flexible, customizable and accurate. The GSM modem was configured, tested and implemented as part of the tracking system to monitor the vehicle's location via SMS and online on Google map. To display the position on Google map, Google Maps API on the Android platform has been used. The Arduino is the brain of the system and the GSM modem is controlled by AT commands that enable data transmission over GSM network while the GPS provide the location data. Whenever the GPS receives a new data it is updated in the database and hence the location can be viewed on Google map.

The successes achieved by the implementation of this project are based on whether the system has achieved its functionality. Therefore, I can humbly say that my system has met all the requirements as specified:

1. The GPS antenna is able to sense GPS location coordinates when powered on and send them to the SIM 808 module for transmission to the server and user via SMS upon request.
2. The Arduino and SIM 808 module is able to establish a GPRS connection with the server and send location data to the server.
3. The system is able to plot the current location on Google Maps and display it to the user on an Android smartphone.
4. The system can send location data via SMS to the user on request. The SMS body also contains a built Google Maps link which when clicked displays the location on the third-party Google Maps app.

Judging by the results, it can be concluded that the goal of this project has been successfully achieved upon completion of this project. The Android application is very easy to use and the tracking device can easily be installed on the vehicle.

Challenges and limitations.

While this Arduino- based tracking system can benefit users, company or any organization, there are also some limitations to using this for vehicle tracking. Often, GPS takes time to connect with the network due to poor weather conditions. For the GPS to work properly, it needs to have a clear view of the sky. That is, it is unlikely to work indoor or may even have problem outside where it has no clear path of transmitting to and receiving signal from satellites. Therefore, due to obstacles like tall buildings or such infrastructure which block view of the sky, often causes multipath error to the receiving signal of the GPS receiver. As a result, location seems to appear to jump from one place to another leading to inaccurate results. Thus, incorrect values of latitude and longitude are sent to the server, for displaying in the Google map on error being initialized.

Recommendations for further work.

This project has demonstrated that IoT-based tracking is technically feasible for fleet management and vehicle tracking. Future projects can explore the following:

a) Reduction of the size of the tracking device by using GPS+GSM on the same module.
b) With the help of high sensitivity vibration sensors, accident detection can be integrated with the system. Upon detection, the system should send the location to the owner, nearest hospitals or the police.
c) Upon reporting the vehicle stolen, the owner can be able to lock the car remotely and disable the engine to avoid further movement.

References.

1. Bett, H. K. (2012). GPS tracking technology adoption in the motor vehicle insurance sector in Kenya (Doctoral dissertation, University of Nairobi, Kenya).
2. Bharath, C.H., Kumar, P.V.,Reddy, P.A. &Abishek, B. (2013). Vehicle tracking system using GSM and GPS (SAGAR Institute of Technology).
3. Baburaokodavati, V.K Raju , S.SrinivasaRao, A.V Prabhu, Dr. Y.V Narayan GSM AND GPS BASED VEHICLE LOCATION AND TRACKING SYSTEM, www.ijera.com(Magazine)
4. Abid Khan ,Ravi Mishra, GPS-GSM based tracking system, International journal of engineering trends and technology-Volume 3Issue 2-2012
5. Marchet, G., Perego, A., & Perotti, S. (2009). An exploratory study of ICT adoption in the Italian freight transportation industry. International Journal of Physical Distribution & Logistics Management, 39(9), 785-812.
6. Ambade Shruti Dinkar and S.A Shaikh, Design and Implementation of Vehicle Tracking System Using GPS, Journal of Information Engineering and
7. Applications, ISSN 2224-5758, Vol 1, No.3, 2011.
8. Asaad M. J. Al-Hindawi, Ibraheem Talib, “Experimentally Evaluation of GPS/GSM Based System Design”, Journal of Electronic Systems Volume 2 Number 2 June 2012.
9. Sonawane, S. GSM and GPS based Real Time Vehicle Tracking System.
10. Dhup, A. Vehicle tracking using the Global Positioning System and the global system for mobile technology.
11. Vehicle tracking system. En.wikipedia.org. (2020). Retrieved 23 November 2020, from https://en.wikipedia.org/wiki/Vehicle_tracking_system.
12. Software Engineering | Iterative Waterfall Model - GeeksforGeeks. (2020). Retrieved 23 November 2020, from: https://www.geeksforgeeks.org/software-engineering-iterative-waterfall model/#:~:text=The%20iterative%20waterfall%20model%20provides,from%20the%20classical%20waterfall%20model.&text=When%20errors%20are%20detected%20at,by%20programmers%20during%20some%20phase.
13. Vehicle Tracking System Based on GPS and GSM. (2020). Retrieved 23 November 2020, from https://create.arduino.cc/projecthub/muchika/vehicle-tracking-system-based-on-gps-and-gsm-57b814
14. Arduino Uno. (2020). Retrieved 23 November 2020, from https://en.wikipedia.org/wiki/Arduino_Uno
15. SIM808 | SIMCom | smart machines, smart decision | simcom.ee. (2020). Retrieved 23 November 2020, from https://simcom.ee/modules/gsm-gprs-

gnss/sim808/#:~:text=SIM808%20module%20is%20a%20complete,to%20develop%20GPS%20enabled%20applications.

16. Aqeel, A. (2020). Introduction to Arduino IDE - The Engineering Projects. Retrieved 23 November 2020, from https://www.theengineeringprojects.com/2018/10/introduction-to-arduino-ide.html
17. Information, U., & System, T. (2020). The Advantages of GPS Tracking System | GPSWOX. Retrieved 23 November 2020, from https://www.gpswox.com/en/blog/useful-information/the-advantages-of-gps-tracking-system
18. Web.archive.org. 2021. What is GPS? Everyday Mysteries. [online] Available at: <https://web.archive.org/web/20180131184150/http://www.loc.gov/rr/scitech/mysteries/global.html> [Accessed 15 February 2021].
19. Sites.google.com. 2021. Understanding NMEA - VMAC GPS/GSM. [online] Available at: <https://sites.google.com/site/vmacgpsgsm/understanding-nmea> [Accessed 15 February 2021].
20. Gakstatter, E. and Gakstatter, E., 2021. What Exactly Is GPS NMEA Data? [online] GPS World. Available at: <https://www.gpsworld.com/what-exactly-is-gps-nmea-data/> [Accessed 15 February 2021].
21. GeeksforGeeks. 2021. *Software Engineering | Iterative Waterfall Model - GeeksforGeeks*. [online] Available at: <https://www.geeksforgeeks.org/software-engineering-iterative-waterfall-model/> [Accessed 19 May 2021].
22. Antratek.com. 2021. *Arduino Uno Rev3*. [online] Available at: <https://www.antratek.com/arduino-uno> [Accessed 19 May 2021].
23. Faranux Electronics. 2021. *SIM808 GSM/GPRS/GPS Bluetooth Compatible Development Board With GPS Antenna COM45 - Faranux Electronics*. [online] Available at: <https://www.dev.faranux.com/product/sim808-gsm-gprs-gps-bluetooth-compatible-development-board-with-gps-antenna-com/> [Accessed 19 May 2021].
24. Acoptex.com. 2021. *Basics: Project 053d SIM808 GSM GPRS GPS Bluetooth evolution board (EVB-V3.2) at Acoptex.com / ACOPTEX.COM*. [online] Available at: <https://acoptex.com/project/264/basics-project-053d-sim808-gsm-gprs-gps-bluetooth-evolution-board-evb-v32-at-acoptexcom/> [Accessed 19 May 2021].
25. Osioplung.com. 2021. *SIM808 EVB-V3.2 Module GSM GPRS GPS Development Board SMA GPS Antenna Arduino*. [online] Available at: <https://www.osioplung.com/index.php?main_page=product_info&products_id=409996> [Accessed 19 May 2021].
26. Youtube.com. 2021. *SIM808 GPS / GSM Locator with Arduino*. [online] Available at: <https://www.youtube.com/watch?v=DeW1LN4BdYU> [Accessed 19 May 2021].

APPENDICES

APPENDIX A: Project Gannt Chart

Table 11: Project Gantt chart

Task name	Start date	End date	Duration	Finish	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Preparation and planning	**01/11/2020**	**27/11/2020**	**27 days**	27/11/2020	🟩	🟩	🟩	🟩															
Develop Project proposal	**01/11/2020**	22/11/2020	22 days	23/11/2020	🟩	🟩	🟩	🟩															
Discuss Project details with Supervisor	22/11/2020	25/11/2020	3 days	25/11/2020				🟩															
Correct proposal as advised by the supervisor	25/11/2020	27/11/2020	2 day	27/11/2020				🟩															
Requirement gathering and analysis	27/11/2020	11/12/2020	44 days	12/01/2021				🟩	🟩	🟩	🟩	🟩											
Eliciting requirements	27/11/2020	12/12/2020	14 days	11/12/2020				🟩	🟩														
Analyse requirements	12/12/2020	19/12/2020	7 days	20/12/2020					🟩														
Evaluate system for feasibility	20/12/2020	26/12/2020	6 days	3/01/2021					🟩														
Requirements modelling	26/12/2020	05/01/2021	10 days	05/01/2021						🟩													
Document requirements	05/01/2021	12/01/2021	7 days	12/01/2021						🟩													
System Design	12/01/2021	02/02/2021	20 days	05/02/2021						🟩	🟩	🟩											
Implementation	02/02/2021	15/04/2021	69 days										🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		
Integration and testing	15/04/2021	22/04/2021	7 days																		🟩		
System documentation	**22/04/2021**	**06/05/2021**	14 days																			🟩	🟩

APPENDIX B: Project Schedule

Table 12: Project Schedule

Task No.	Task Name	Planned Days	Actual days	Start day	Actual start day	End day	Actual end day	Deliverables
1	Develop Project proposal	22	23	**01/11/2020**	**01/11/2020**	22/11/2020	23/11/2020	Summary of what has been done in the field of research and what's to be developed.
2	Discuss Project details with Supervisor	3	3	22/11/2020	23/11/2020	25/11/2020	25/11/2020	Key changes to the project proposal.
3	Correct proposal as advised by the supervisor	2	1	25/11/2020	26/11/2020	27/11/2020	27/11/2020	A refined project proposal as required by the supervisor.
4	Eliciting requirements	14	14	27/11/2020	28/11/2020	12/12/2020	11/12/2020	Determine what requirements will consist of and document.
5	Analyse requirements	7	8	12/12/2020	12/12/2020	19/12/2020	20/12/2020	Determine whether the stated requirements are unclear, incomplete, ambiguous, or contradictory
6	Evaluate system for feasibility	6	12	20/12/2020	21/12/2020	26/12/2020	3/01/2021	Determine technical and operational

								feasibility and document
7	Requirements modelling	10	2	26/12/2020	03/01/2021	05/01/2021	05/01/2021	Requirement Specification document.
8	Document requirements	7	7	05/01/2021	06/01/2021	12/01/2021	12/01/2021	Requirement Specifications, Use cases, data flow diagram, context diagrams, functional decomposition diagram.
9	**System Design**	20	23	12/01/2021	12/01/2021	02/02/2021	05/02/2021	System architecture, network architecture, database design, user interface design.
10	**Implementation**	69	69	02/02/2021	05/02/2021	15/04/2021	15/04/2021	Source code
11	Integration and testing	7	7	15/04/2021	15/04/2021	22/04/2021	22/04/2021	
12.	Document the system	14	18	**22/04/2021**	**22/04/2021**	**06/05/2021**	**10/05/2021**	Project report

APPENDIX C: Project Budget.

Table 13: Project Budget

Resource	Cost (Kshs)
Journals/ Research papers	5000
Internet sources	500
Arduino Uno R3 microcontroller.	1500
SIM 808 with GPS and GSM modules.	2600
Power supply (Size AA Battery holder and three dry cell batteries)	300
LED	5
LCD (16X2)	500
Connecting Wires	300
Software (Google Maps, Arduino IDE and Android Studio)	Free
Total	10705

APPENDIX D: MOBILE APPLICATION USER MANUAL.

Installing the IoT hardware.

The below diagram details the connection of the components:

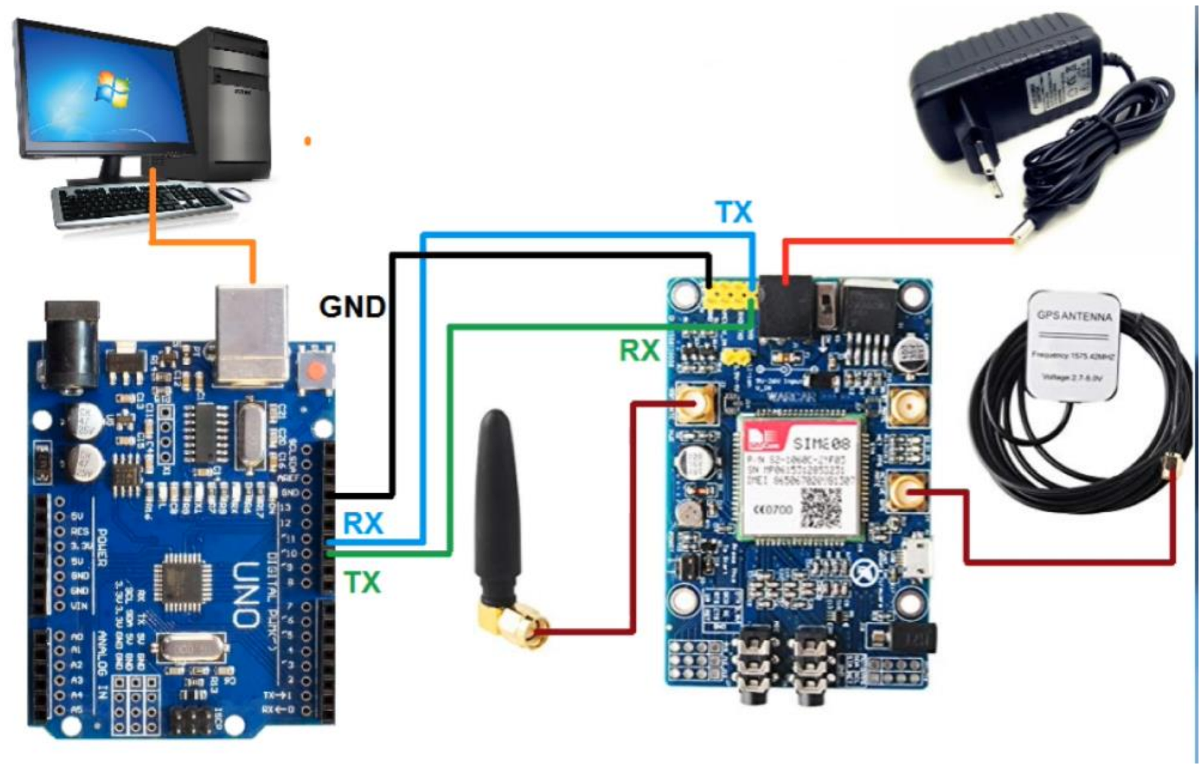


Figure 24: Connection of the Arduino IoT components, Source: (SIM808 GPS / GSM Locator with Arduino, 2021)

Once the Arduino is powered on, it attempts to prepare the GSM module for use by sending a number of AT commands to initialise it. If successful, the SIM 808 GPS Module begins searching for a GPS fix from available satellites. The GPS module forwards the data to the Arduino where it is timestamped and cached. A JSON document is compiled from the contents of the cache and sent to the GSM module as part of web request command. If the HTTP request is successful, returning a 200 or a 201, the cache is cleared.

Installing the mobile app.

System requirements.

The App can run on any Android smartphone having Android Version 4.4 and above. A smartphone with 512MB of RAM is sufficient. Google Play must be installed on the device in order to support Google Maps. The app runs only when there is an internet connection.

Navigation guide.

When the user clicks on the app icon, the splash screen shown below shows for some few seconds. The user is then prompted to select to either log in or create an account

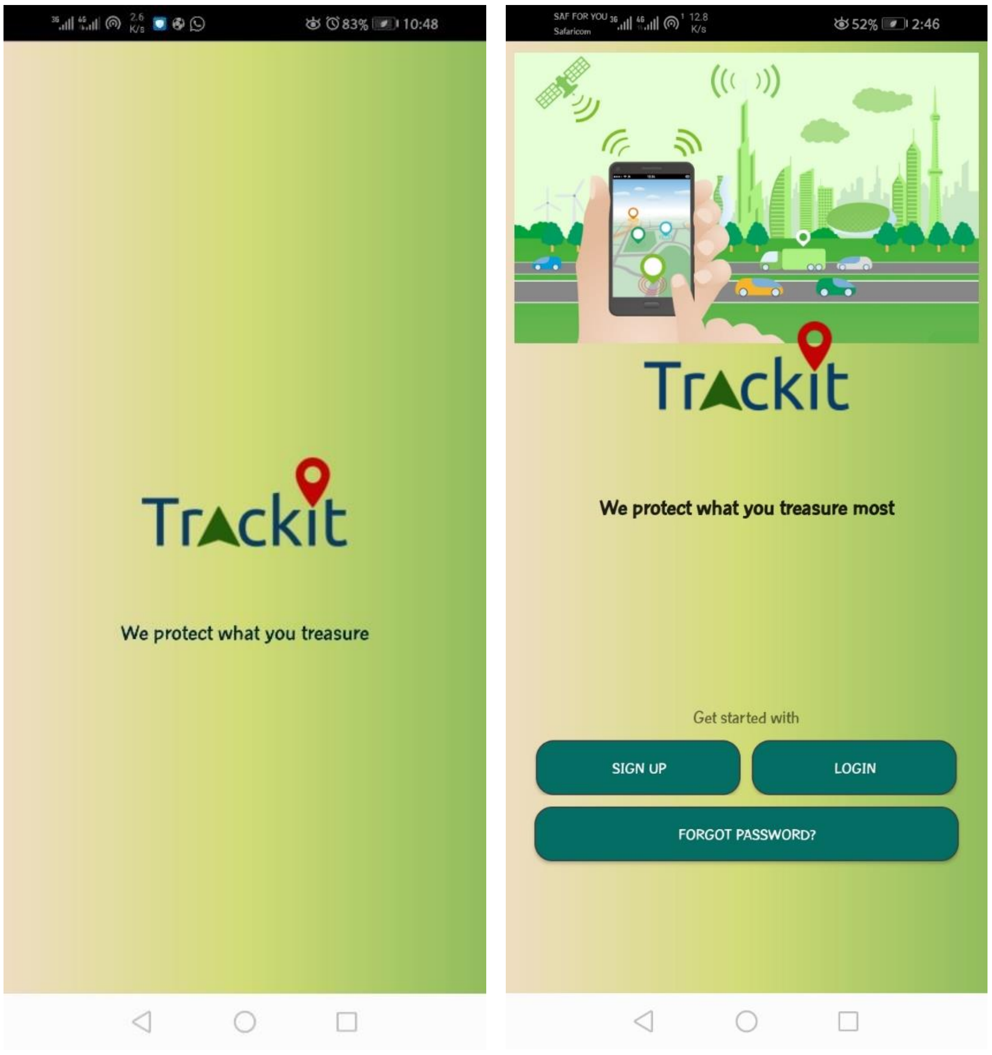
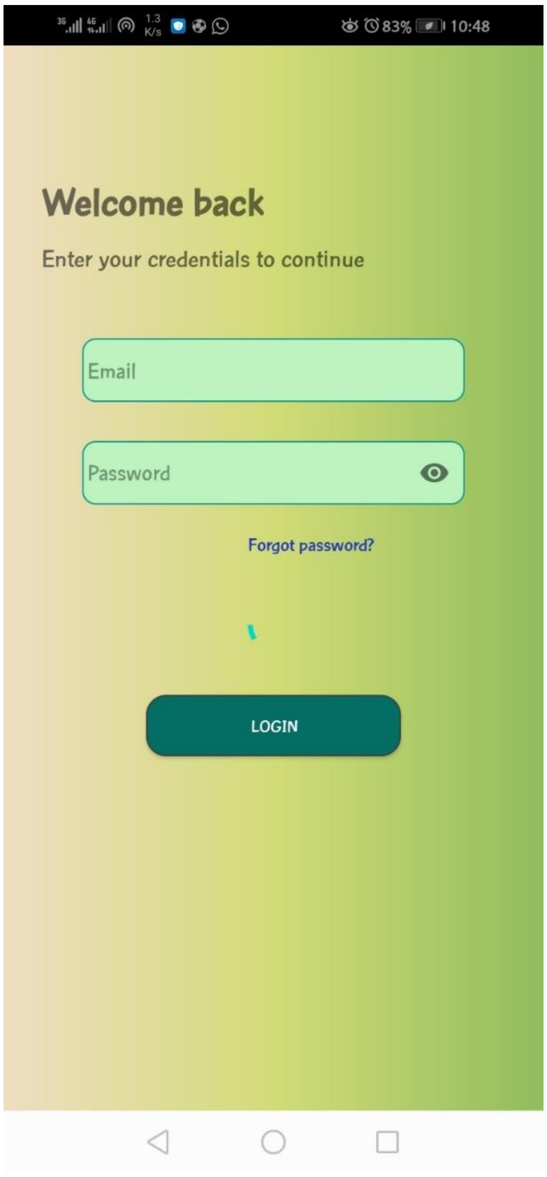


Figure 25: Splash screen and landing page

The log in page asks the user of email and password. The process of creating an account asks the user of personal details first, then vehicle details and then lastly the tracking device details.



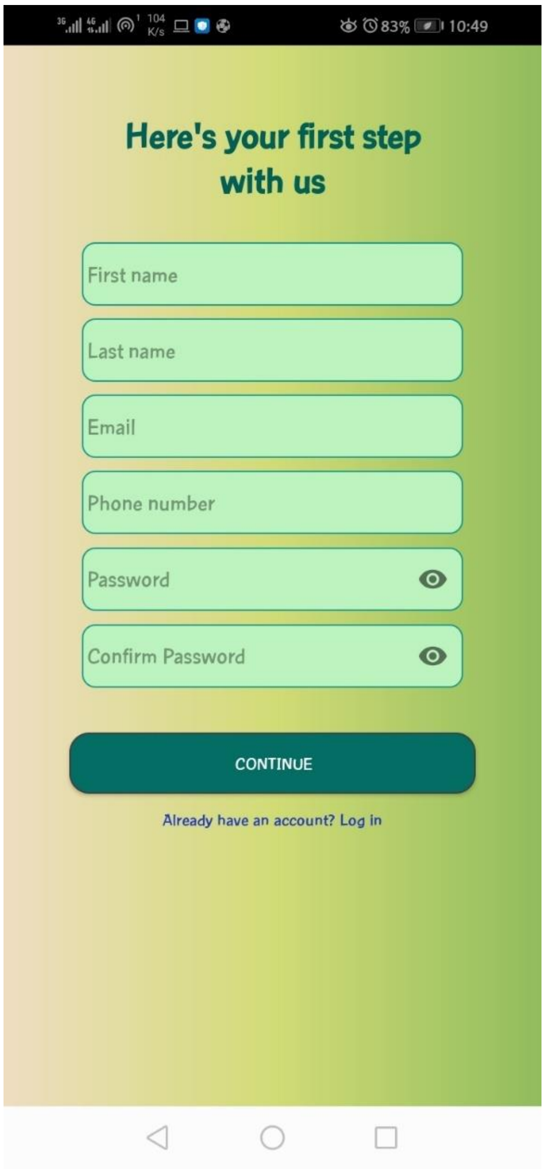


Figure 26: LogIn page and create account page

Upon Log in, the user is taken to a homepage which shows the current location of his/her car in Google Maps.

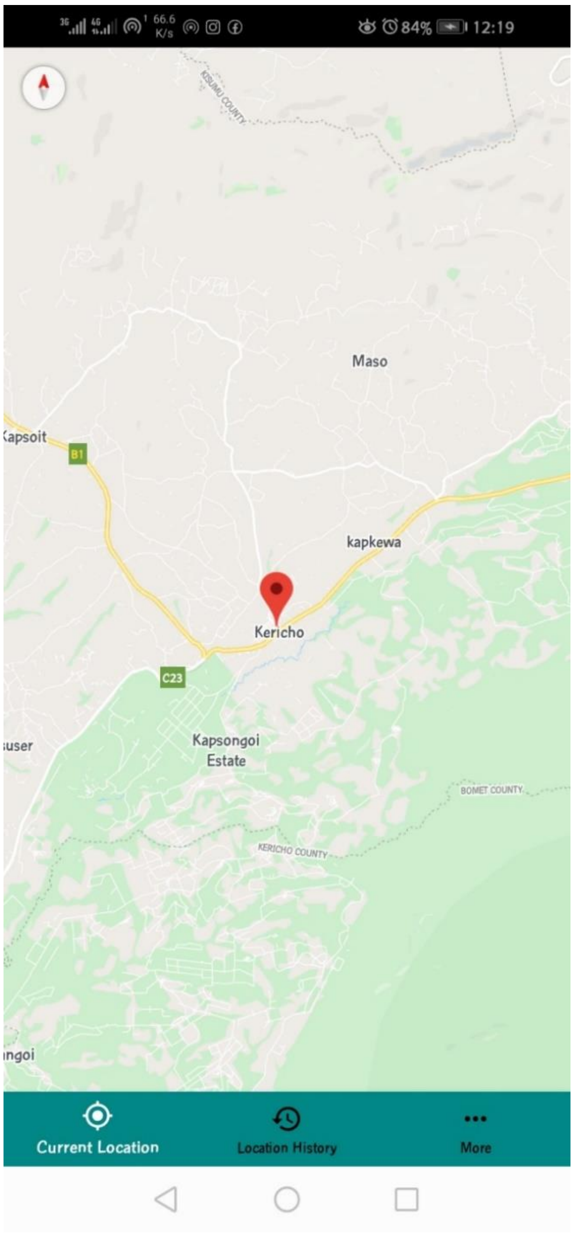


Figure 27: Location map activity showing car's current location

The user can select to view location history or go to account settings/More. If the user selects Location History, then he/she is prompted to select a date from a date picker dialog for which to show the location history of the car in that period.

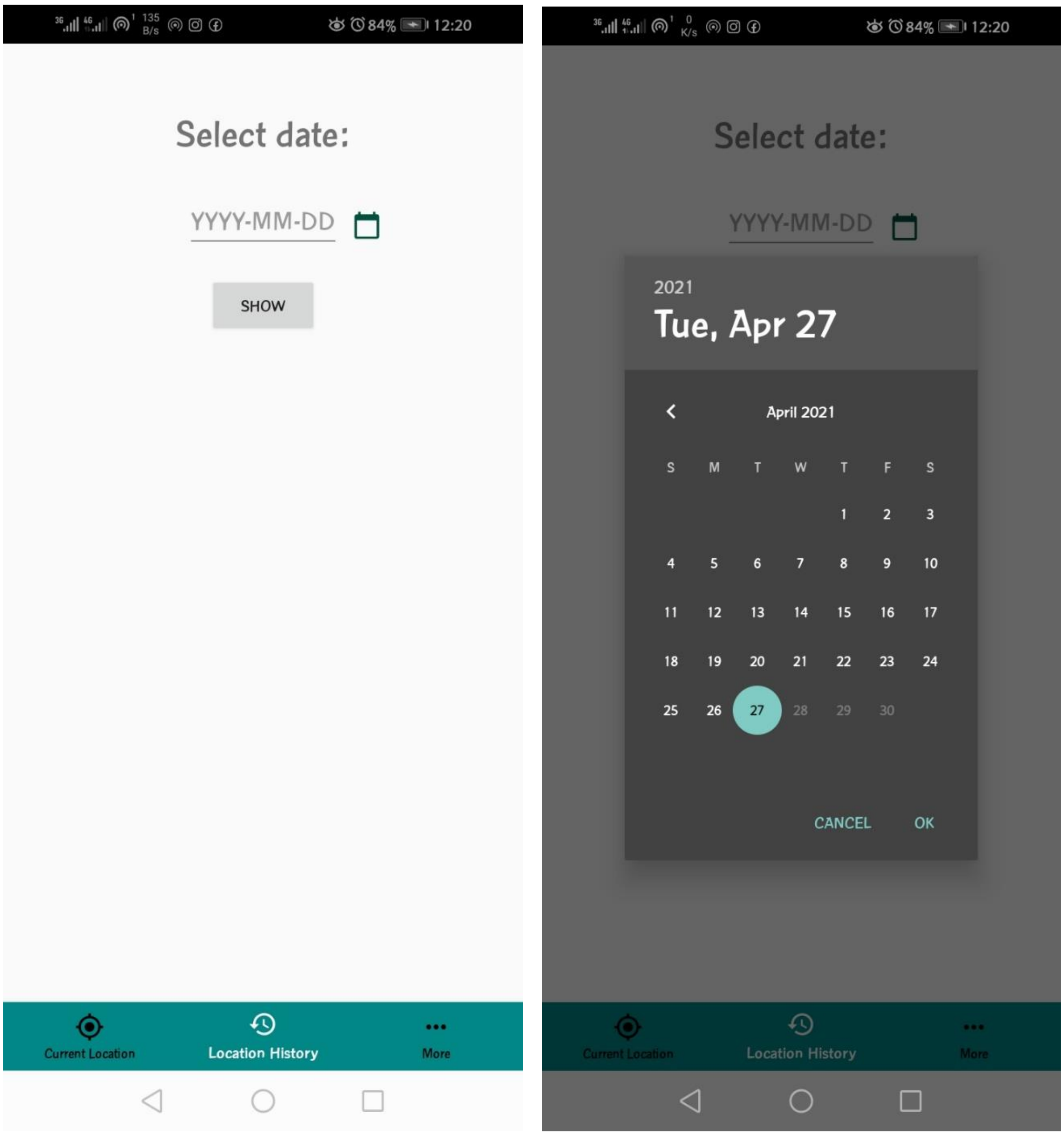


Figure 28: Location History activity to prompt user to select a date to show location history

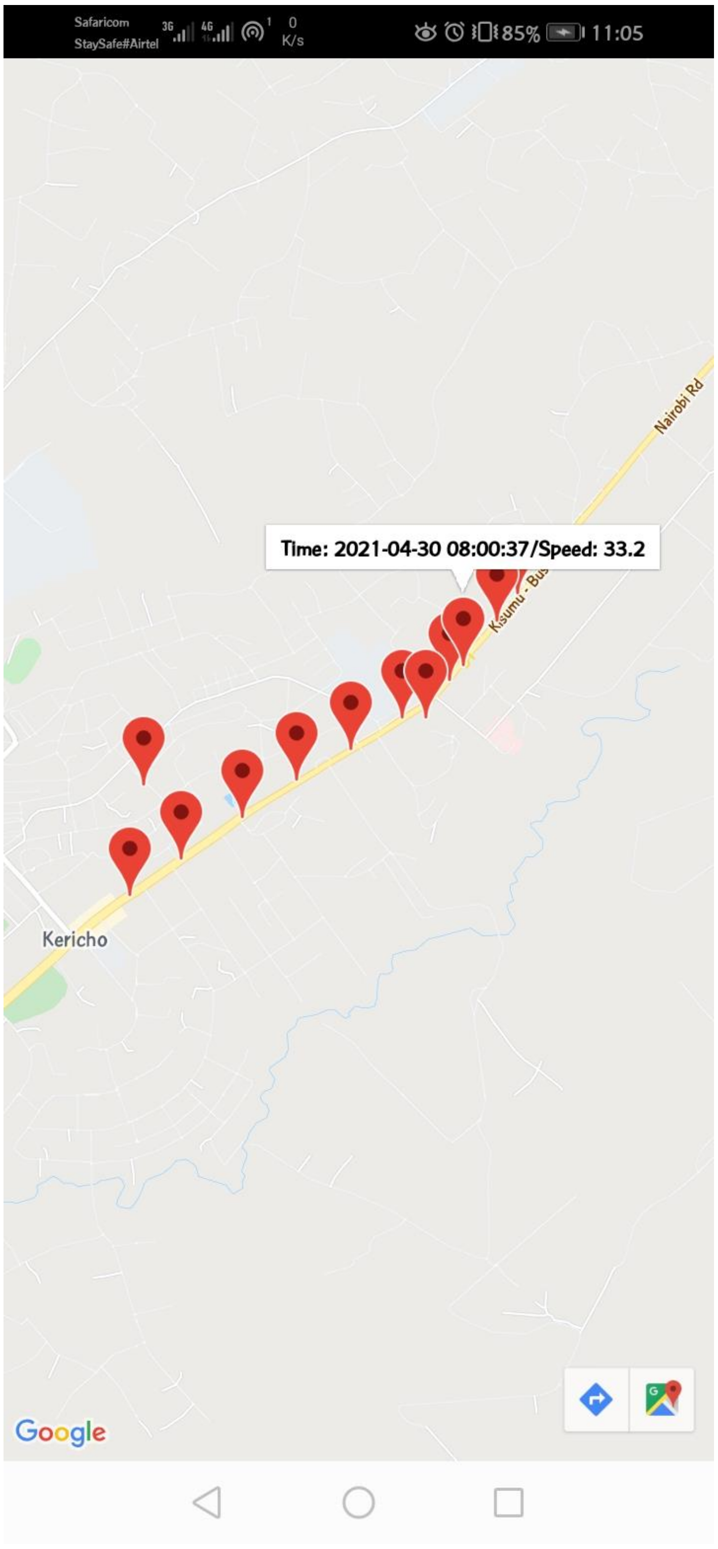


Figure 29: Location history page showing previous locations during the date selected

When the user selects More, then he/she is taken to a page with more options i.e. My account, my car, report a stolen car, About the tracking device, help in installing tracking device and a Logout button.

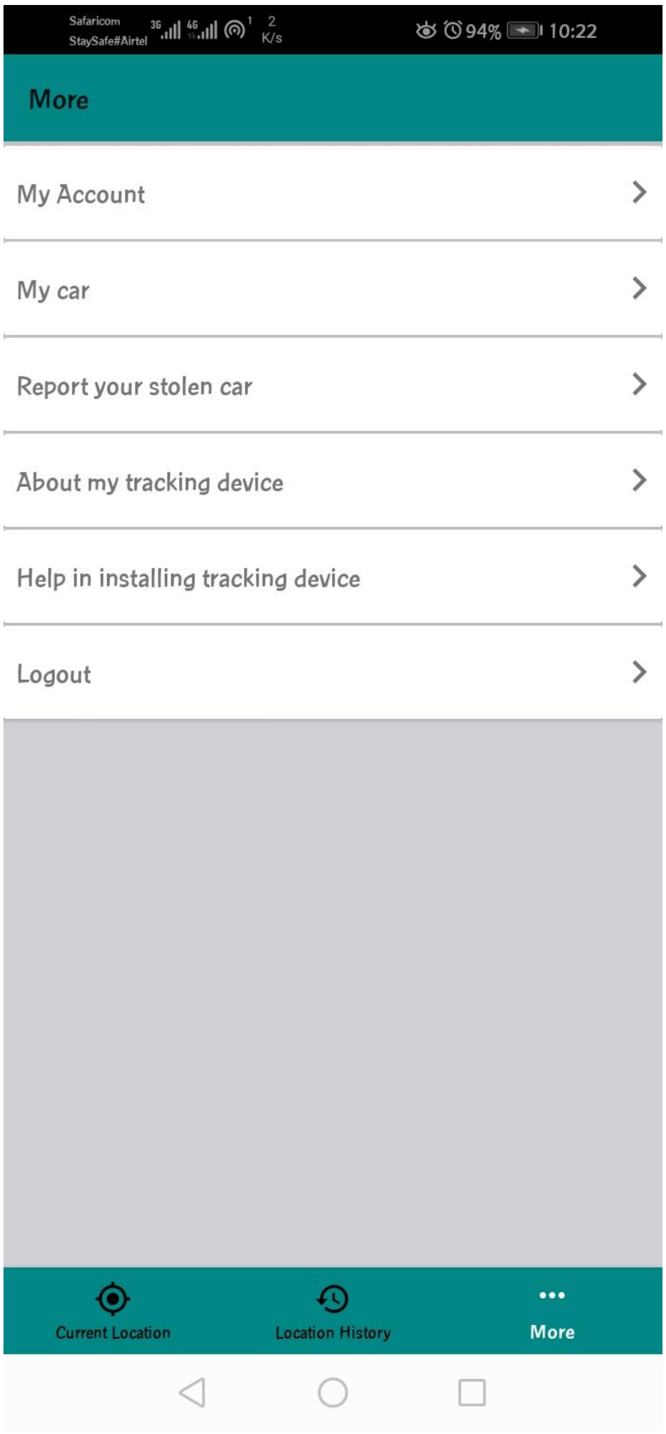
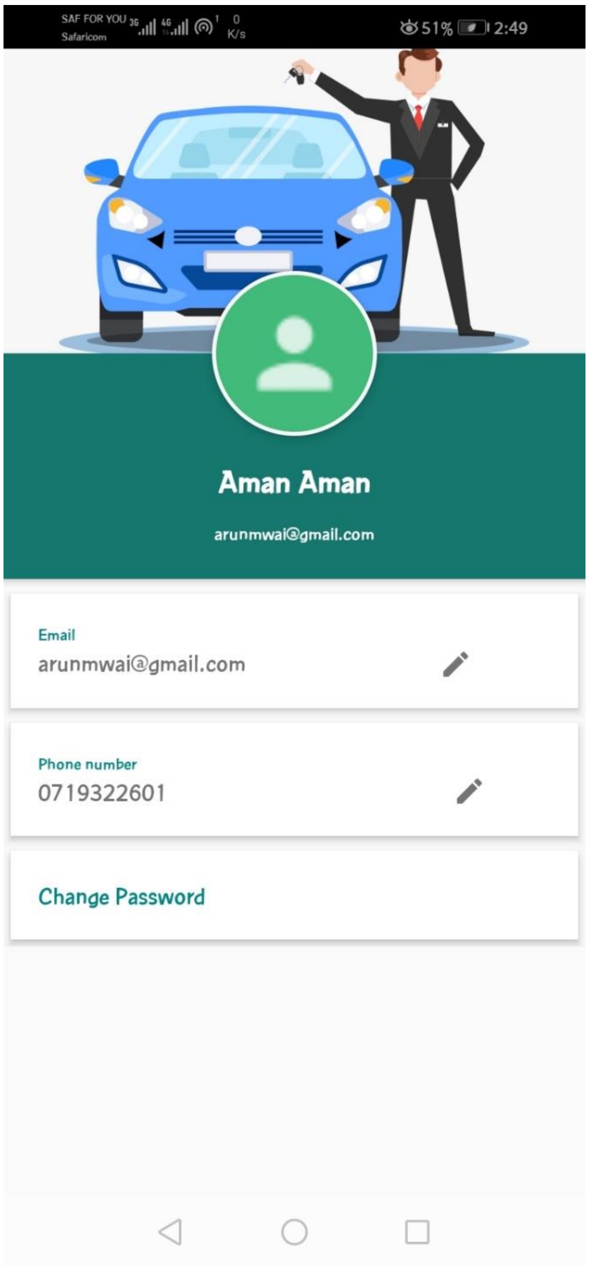


Figure 30: More activity showing more options

On the My Account page, the user views his/her Name, Phone number, Email. He/she is able to change the details with the click of the buttons having the edit icon. The user is able to change his/her account password on this page also.



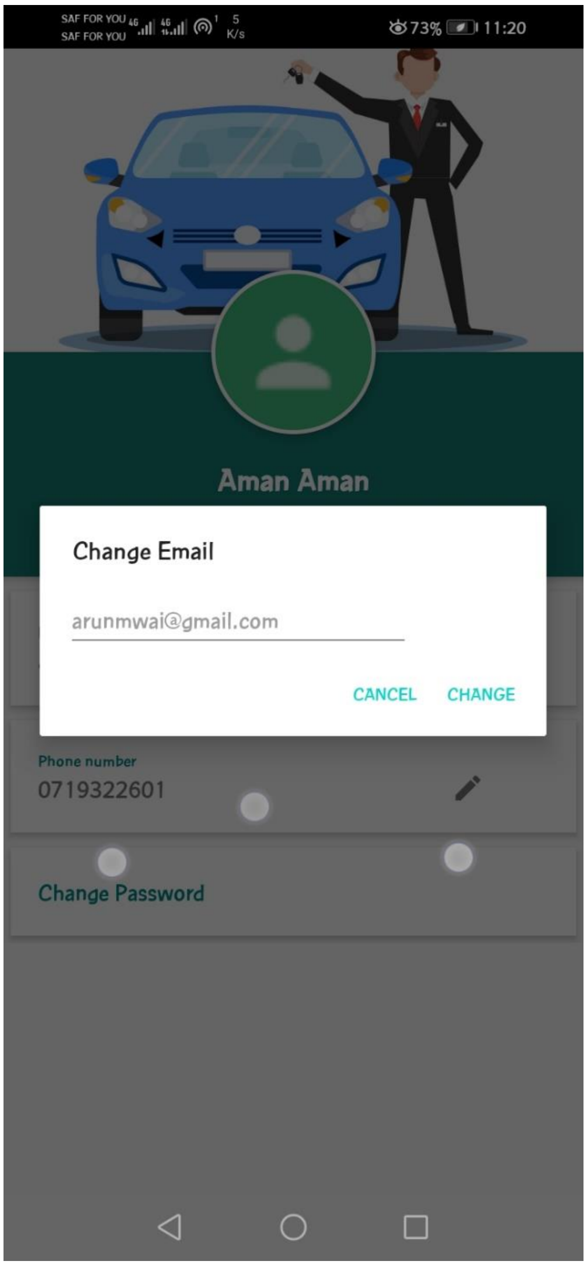
Aman Aman

arunmwai@gmail.com

Email
arunmwai@gmail.com

Phone number
0719322601

Change Password



Aman Aman

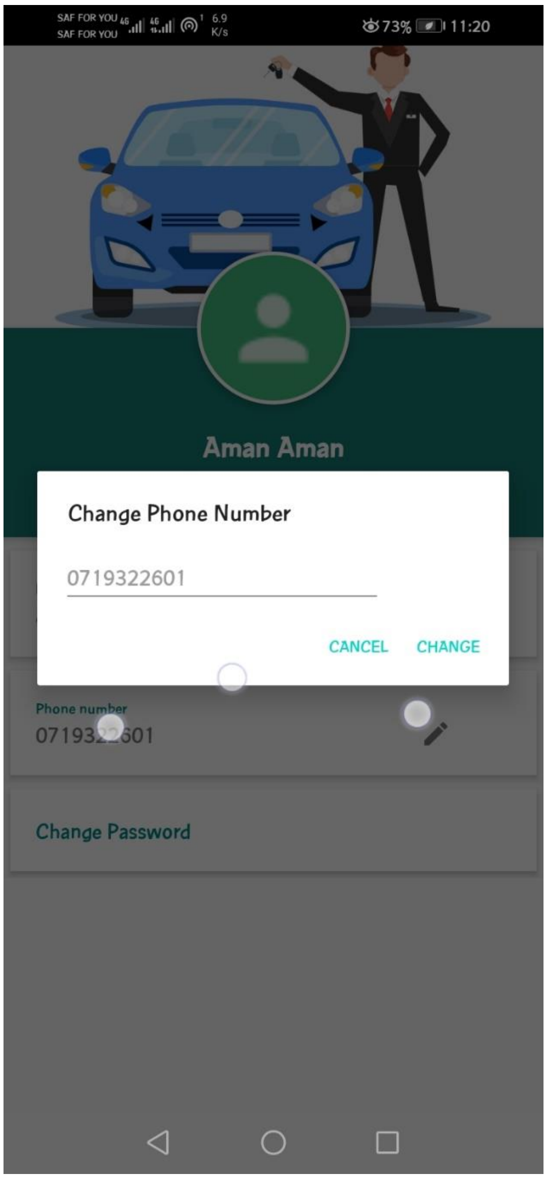
Change Email

arunmwai@gmail.com

CANCEL CHANGE

Phone number
0719322601

Change Password



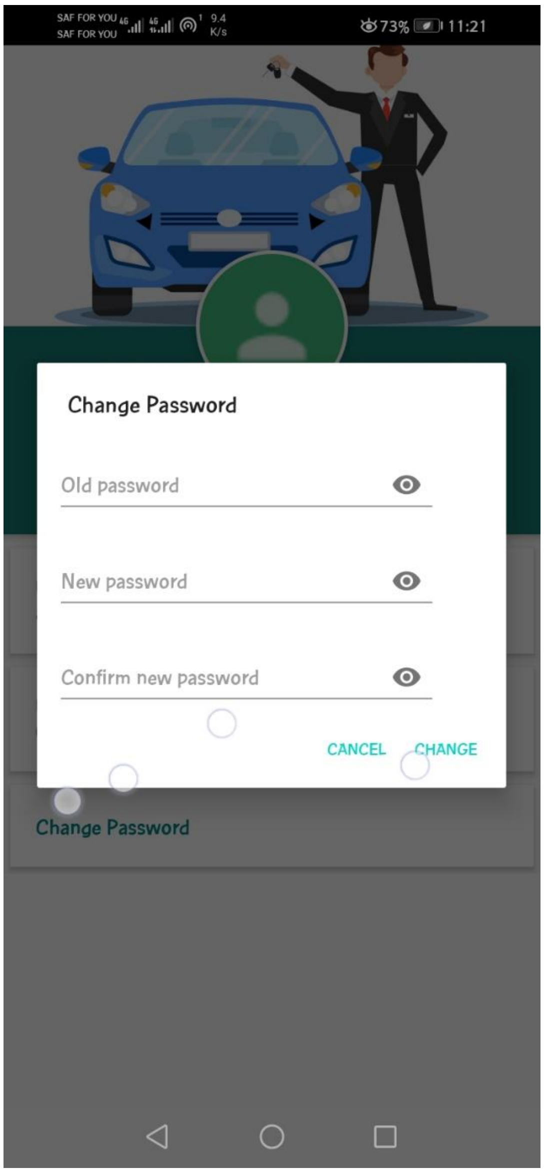
Change Phone Number

0719322601

CANCEL CHANGE

Phone number
0719322601

Change Password



Change Password

Old password

New password

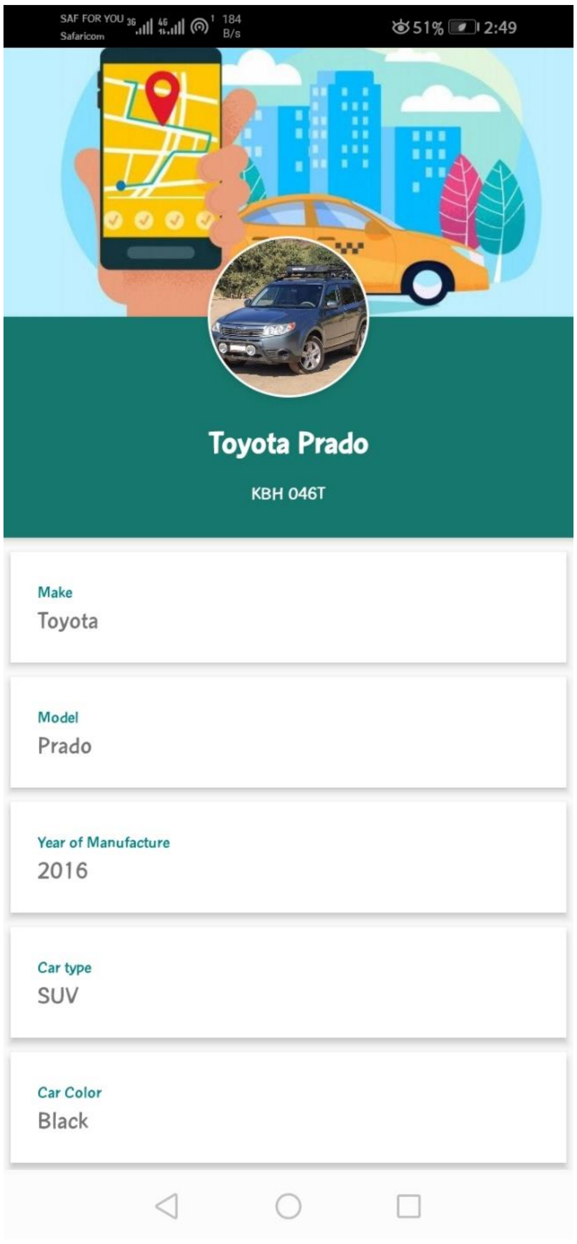
Confirm new password

CANCEL CHANGE

Change Password

Figure 31: My account activity and change profile options

On the My Car page, the user can view the vehicle details input during registration.



Toyota Prado

KBH 046T

Make
Toyota

Model
Prado

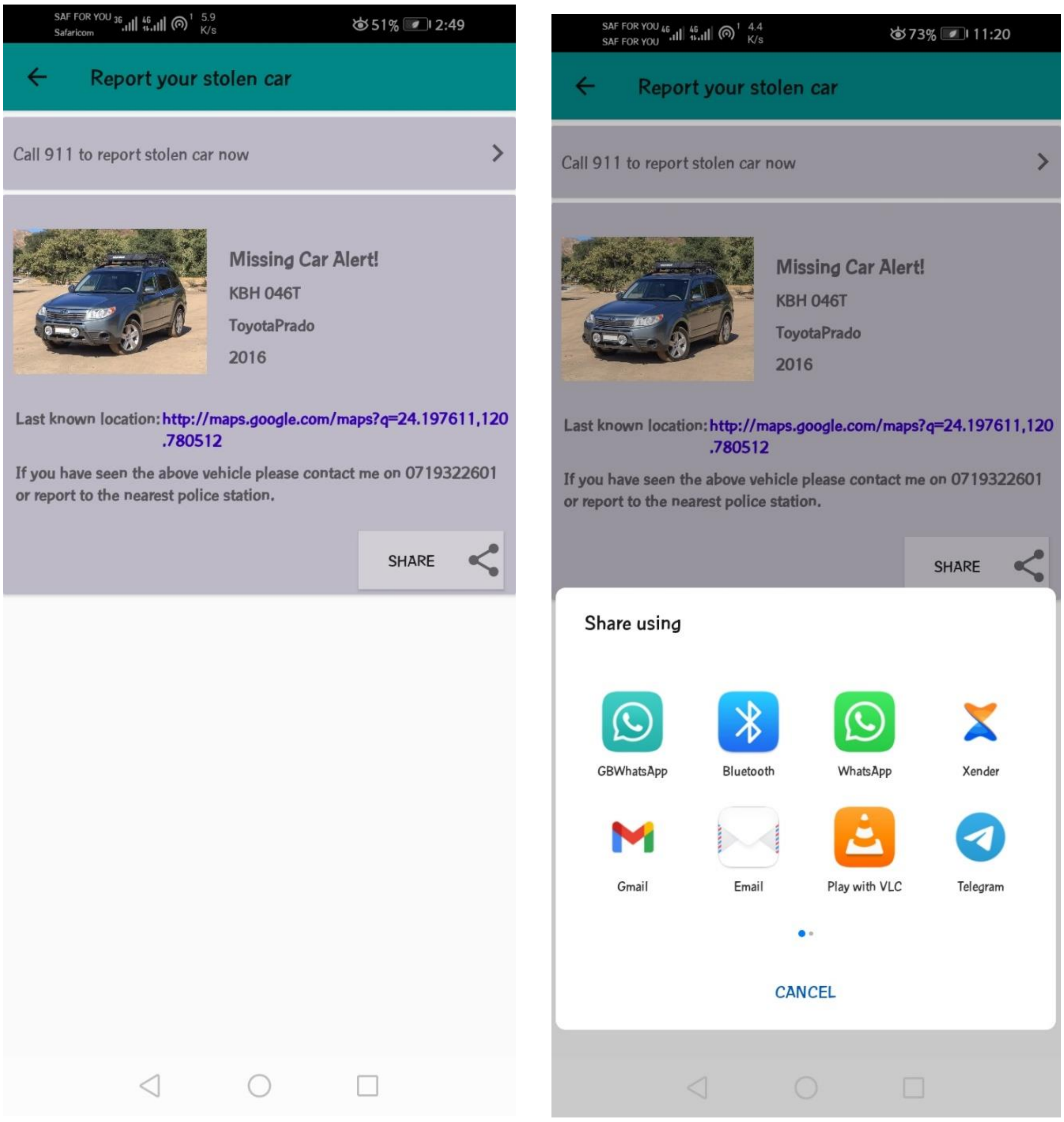
Year of Manufacture
2016

Car type
SUV

Car Color
Black

Figure 32: My car activity showing car details

On the Report Car Stolen Page, the user can select to either call the police hotline or share the alert shown as a message on social media.


Report your stolen car

Call 911 to report stolen car now

Missing Car Alert!
KBH 046T
ToyotaPrado
2016

Last known location: http://maps.google.com/maps?q=24.197611,120.780512

If you have seen the above vehicle please contact me on 0719322601 or report to the nearest police station.

SHARE

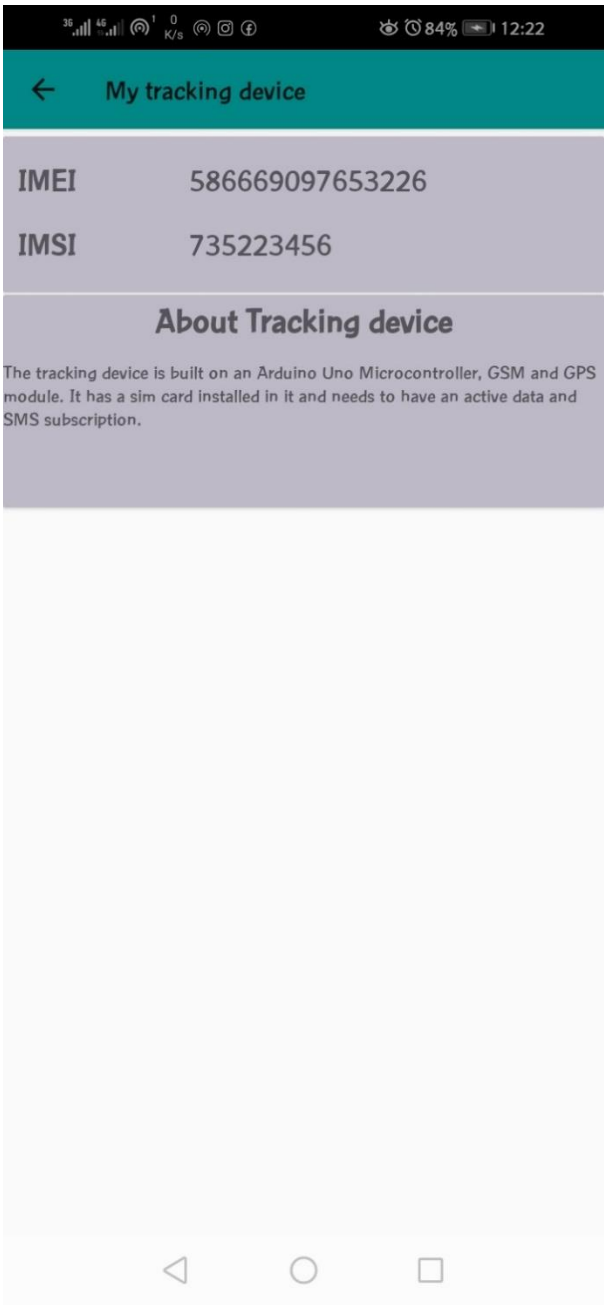
Share using

GBWhatsApp, Bluetooth, WhatsApp, Xender, Gmail, Email, Play with VLC, Telegram

CANCEL

Figure 33: Report stolen car page and on-click of the share button respectively

On the About My tracking device details, the user can view the details of the tracking device as were put during the registration process.



My tracking device

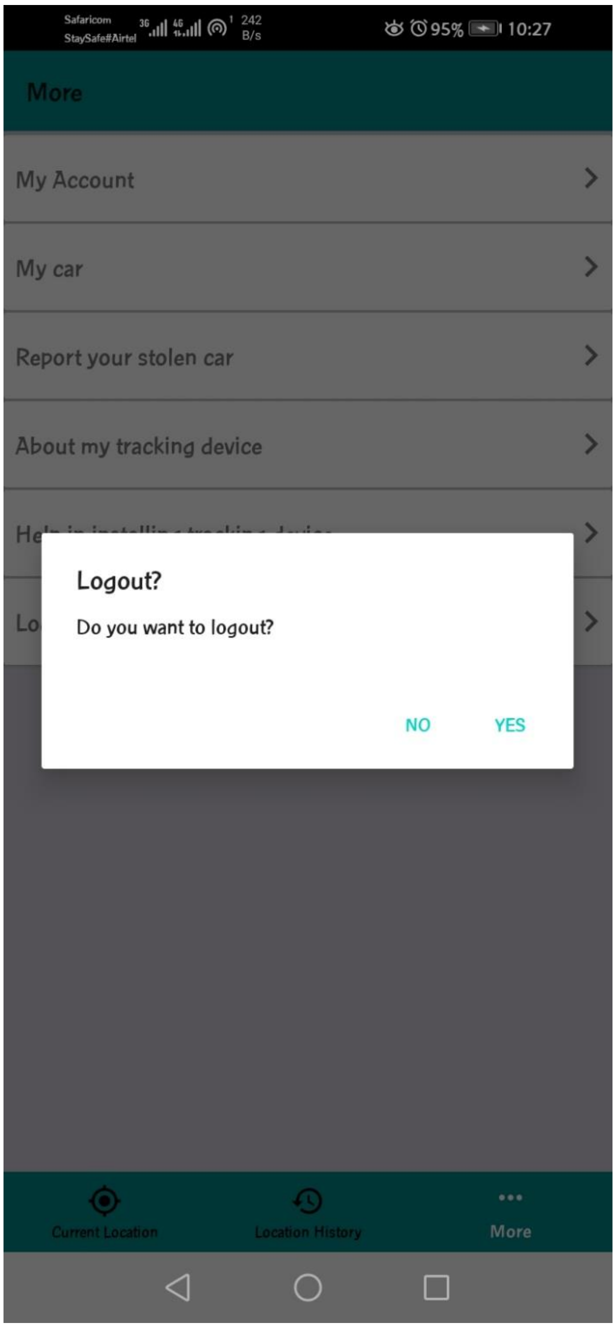
IMEI	586669097653226
IMSI	735223456

About Tracking device

The tracking device is built on an Arduino Uno Microcontroller, GSM and GPS module. It has a sim card installed in it and needs to have an active data and SMS subscription.

Figure 34: Tracking device details page

The user can log out by selecting the log out option in the More page.


Logout?

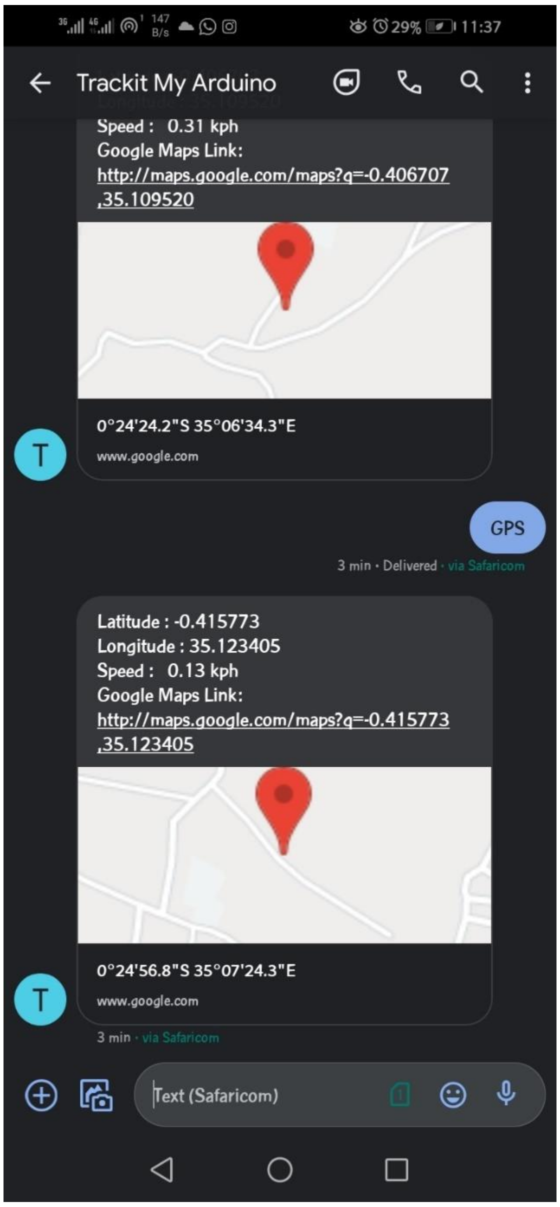
Do you want to logout?

Figure 35: Logout prompt

Getting the location details via SMS

The user also has the option to get location details via SMS. This feature was added to enhance efficiency and availability in areas where there is no GPRS to transfer the location data to the web server. The user should send an SMS to the number inserted in the tracking device and the SMS message body should be “GPS”. Either case is okay because the comparison is case insensitive.

The below example screenshot shows the above:



Trackit My Arduino

Speed : 0.31 kph
Google Maps Link:
http://maps.google.com/maps?q=-0.406707,35.109520

0°24'24.2"S 35°06'34.3"E
www.google.com

GPS

3 min • Delivered • via Safaricom

Latitude : -0.415773
Longitude : 35.123405
Speed : 0.13 kph
Google Maps Link:
http://maps.google.com/maps?q=-0.415773,35.123405

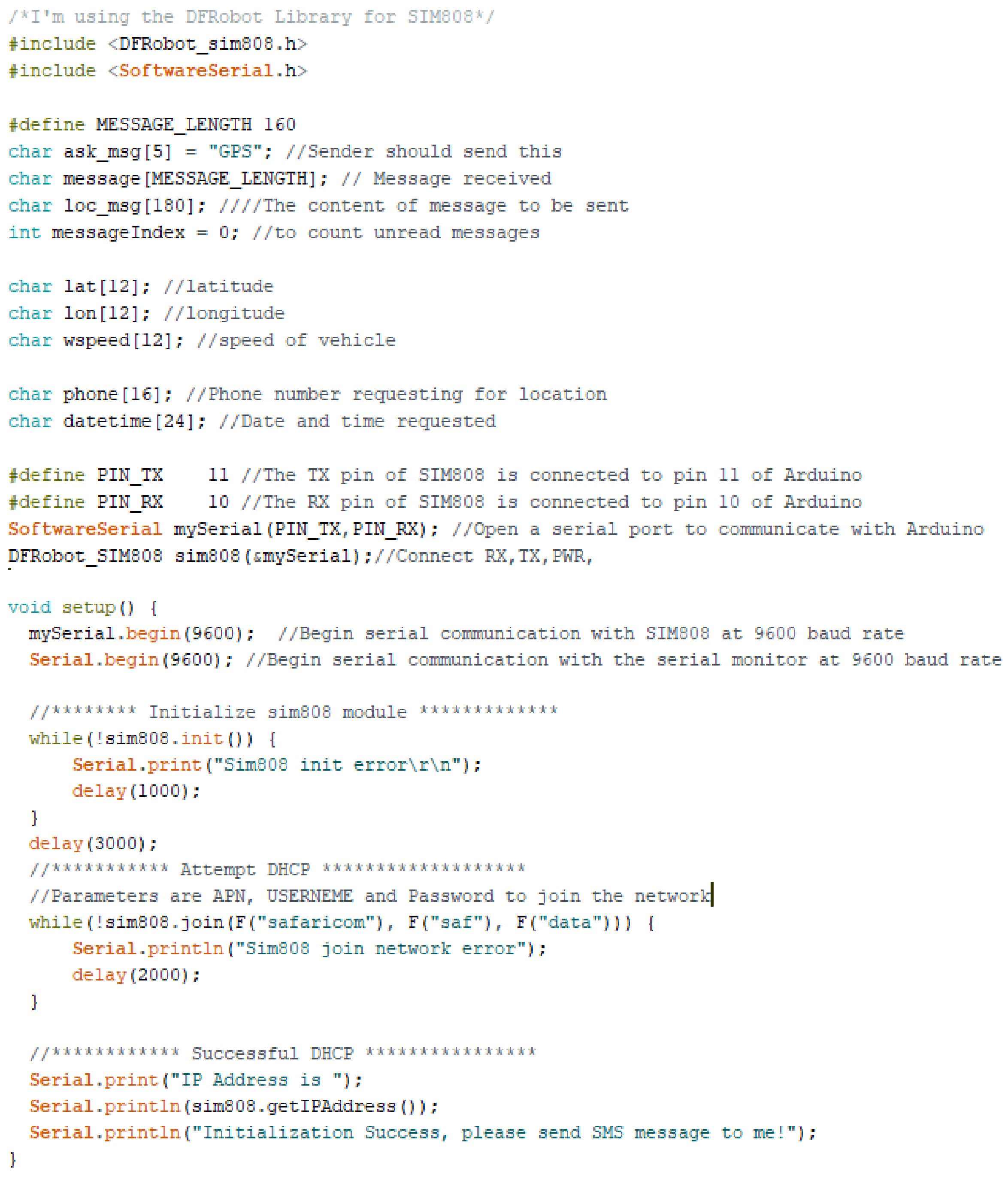
0°24'56.8"S 35°07'24.3"E
www.google.com

3 min • via Safaricom

Figure 36: Screenshot showing request for location via SMS

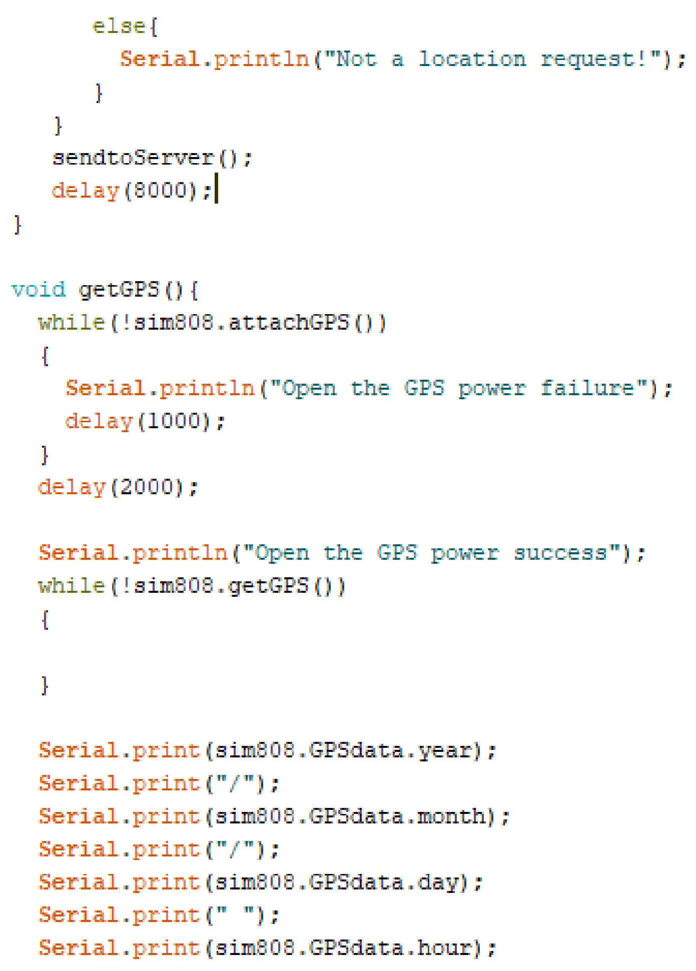
As seen, the tracker sends the location latitude, longitude, speed and a link that can be used to view the location on Google Maps.

APPENDIX E: SAMPLE ARDUINO CODE.

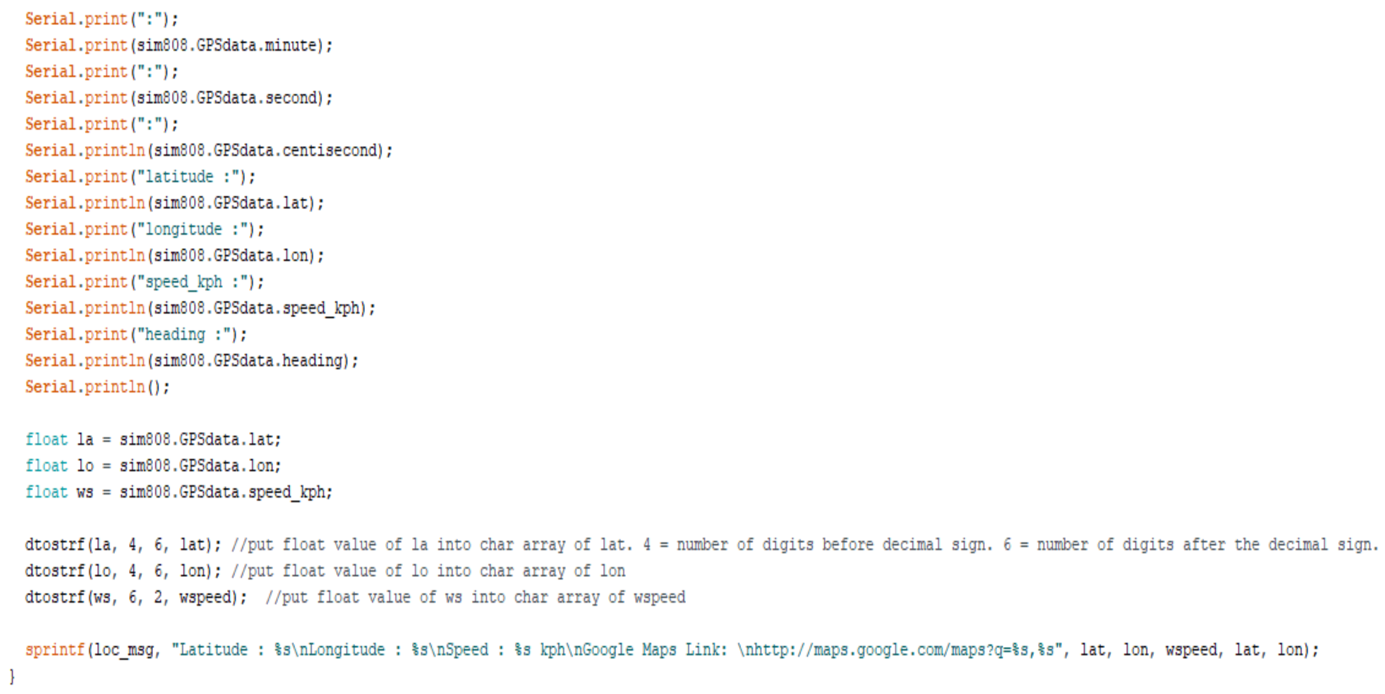


```
/*I'm using the DFRobot Library for SIM808*/
#include <DFRobot_sim808.h>
#include <SoftwareSerial.h>

#define MESSAGE_LENGTH 160
char ask_msg[5] = "GPS"; //Sender should send this
char message[MESSAGE_LENGTH]; // Message received
char loc_msg[180]; ////The content of message to be sent
int messageIndex = 0; //to count unread messages

char lat[12]; //latitude
char lon[12]; //longitude
char wspeed[12]; //speed of vehicle

char phone[16]; //Phone number requesting for location
char datetime[24]; //Date and time requested

#define PIN_TX    11 //The TX pin of SIM808 is connected to pin 11 of Arduino
#define PIN_RX    10 //The RX pin of SIM808 is connected to pin 10 of Arduino
SoftwareSerial mySerial(PIN_TX,PIN_RX); //Open a serial port to communicate with Arduino
DFRobot_SIM808 sim808(&mySerial);//Connect RX,TX,PWR,

void setup() {
  mySerial.begin(9600);  //Begin serial communication with SIM808 at 9600 baud rate
  Serial.begin(9600); //Begin serial communication with the serial monitor at 9600 baud rate

  //******** Initialize sim808 module *************
  while(!sim808.init()) {
      Serial.print("Sim808 init error\r\n");
      delay(1000);
  }
  delay(3000);
  //*********** Attempt DHCP *******************
  //Parameters are APN, USERNEME and Password to join the network
  while(!sim808.join(F("safaricom"), F("saf"), F("data"))) {
      Serial.println("Sim808 join network error");
      delay(2000);
  }

  //************ Successful DHCP ****************
  Serial.print("IP Address is ");
  Serial.println(sim808.getIPAddress());
  Serial.println("Initialization Success, please send SMS message to me!");
}
```

```
void loop() {
  getGPS();

  //*********** Detecting unread SMS ************************
   messageIndex = sim808.isSMSunread();
    Serial.print("messageIndex: ");
    Serial.println(messageIndex);

   //*********** At least, there is one UNREAD SMS ***********
   if (messageIndex > 0) {
      sim808.readSMS(messageIndex, message, MESSAGE_LENGTH, phone, datetime);
      delay(2000);

      //***********In order not to full SIM Memory, is better to delete it**********
      sim808.deleteSMS(messageIndex);
      Serial.print("From number: ");
      Serial.println(phone);
      Serial.print("Datetime: ");
      Serial.println(datetime);
      Serial.print("Received Message: ");
      Serial.println(message);

      String aa = String(ask_msg);
      String bb = String(message);

      if(aa.equalsIgnoreCase(bb)){
        //******** define phone number and text **********
        sim808.sendSMS(phone,loc_msg);
      }
```



```
      else{
        Serial.println("Not a location request!");
      }
   }
   sendtoServer();
   delay(8000);
}

void getGPS(){
  while(!sim808.attachGPS())
  {
    Serial.println("Open the GPS power failure");
    delay(1000);
  }
  delay(2000);

  Serial.println("Open the GPS power success");
  while(!sim808.getGPS())
  {

  }

  Serial.print(sim808.GPSdata.year);
  Serial.print("/");
  Serial.print(sim808.GPSdata.month);
  Serial.print("/");
  Serial.print(sim808.GPSdata.day);
  Serial.print(" ");
  Serial.print(sim808.GPSdata.hour);
```



```
  Serial.print(":");
  Serial.print(sim808.GPSdata.minute);
  Serial.print(":");
  Serial.print(sim808.GPSdata.second);
  Serial.print(":");
  Serial.println(sim808.GPSdata.centisecond);
  Serial.print("latitude :");
  Serial.println(sim808.GPSdata.lat);
  Serial.print("longitude :");
  Serial.println(sim808.GPSdata.lon);
  Serial.print("speed_kph :");
  Serial.println(sim808.GPSdata.speed_kph);
  Serial.print("heading :");
  Serial.println(sim808.GPSdata.heading);
  Serial.println();

  float la = sim808.GPSdata.lat;
  float lo = sim808.GPSdata.lon;
  float ws = sim808.GPSdata.speed_kph;

  dtostrf(la, 4, 6, lat); //put float value of la into char array of lat. 4 = number of digits before decimal sign. 6 = number of digits after the decimal sign.
  dtostrf(lo, 4, 6, lon); //put float value of lo into char array of lon
  dtostrf(ws, 6, 2, wspeed);  //put float value of ws into char array of wspeed

  sprintf(loc_msg, "Latitude : %s\nLongitude : %s\nSpeed : %s kph\nGoogle Maps Link: \nhttp://maps.google.com/maps?q=%s,%s", lat, lon, wspeed, lat, lon);
}
```

```
void sendtoServer(){
  String url;
  url = "GET /storeLocation.php?trackingUnit=5&latitude=";
  url += lat;
  url += "&longitude=";
  url += lon;
  url += "&speed=";
  url += wspeed;
  url += " HTTP/1.1\r\nHost:kiptrack.000webhostapp.com\r\n\r\n";

  int str_len = url.length() + 1;
  char char_array[str_len];
  url.toCharArray(char_array, str_len);

  char buffer[512];

  //*********** Send a GET request *****************
  Serial.println("waiting to fetch...");
  sim808.send(char_array, sizeof(char_array)-1);
  while (true) {
      int ret = sim808.recv(buffer, sizeof(buffer)-1);
      if (ret <= 0){
          Serial.println("fetch over...");
          break;
      }
      buffer[ret] = '\0';
      Serial.print("Recv: ");
      Serial.print(ret);

      Serial.print(" bytes: ");
      Serial.println(buffer);
      break;
  }
}
```

APPENDIX F: SAMPLE ANDROID JAVA CODE.

The code below shows how I added a marker of the current location in Google maps

```java
/**
 * Manipulates the map once available.
 * This callback is triggered when the map is ready to be used.
 * This is where we can add markers or lines, add listeners or move the
camera. In this case,
 * we just add a marker near Sydney, Australia.
 * If Google Play services is not installed on the device, the user will
be prompted to install
 * it inside the SupportMapFragment. This method will only be triggered
once the user has
 * installed Google Play services and returned to the app.
 */
@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;

    // Add a marker in current location and move the camera
    LatLng carLocation = new LatLng(myLatitude, myLongitude);
    mMap.addMarker(new MarkerOptions().position(carLocation).title("Lat: "
+ myLatitude + " Long: " + myLongitude));
    mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(carLocation,
12));
}
```

The below code fetches the current location from the database and returns the latitude and longitude values fetched

```java
//Send a request to the database to get the current location co-ordinates
public void getLocation(){
    final double[] location = new double[2];
    SessionManager sessionManager = new SessionManager(this);
    HashMap<String, String> user = sessionManager.getUserDetail();
    final String URL =
"https://kiptrack.000webhostapp.com/getLocation.php";
    final String trackID = user.get("trackingDeviceID");
    StringRequest stringRequest = new StringRequest(Request.Method.POST,
URL, new Response.Listener<String>() {
        @Override
        public void onResponse(String response) {
            try {
                JSONObject jsonObject = new JSONObject(response);
                String code = jsonObject.getString("Code");
                String message = jsonObject.getString("Message");

                if (code.equals("1")){
                    double latitude = jsonObject.getDouble("latitude");
                    double longitude = jsonObject.getDouble("longitude");
                    myLatitude = latitude;
                    myLongitude = longitude;

                    initializeMap();

                    Toast.makeText(LocationMap.this, message + myLatitude
+","+ myLongitude, Toast.LENGTH_SHORT).show();
                }
                else if ((code.equals("2"))){
                    Toast.makeText(LocationMap.this, message,
Toast.LENGTH_SHORT).show();
                }
                else if (code.equals("3")){

                    Toast.makeText(LocationMap.this, message,
Toast.LENGTH_SHORT).show();
                }

            } catch (JSONException e) {
                e.printStackTrace();
                Toast.makeText(LocationMap.this, "Could not update
location. Try again!" + e.toString(), Toast.LENGTH_SHORT).show();
            }
        }
    }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            Toast.makeText(LocationMap.this, "No update on location.Try
later" + error.toString(), Toast.LENGTH_SHORT).show();
        }
    })
    {
        @Nullable
        @Override
        protected Map<String, String> getParams() throws AuthFailureError
{
            Map<String, String> params = new HashMap<>();
            params.put("trackID", trackID);
            return params;
```

```
private void LoginFunction(final String email, final String password) {
        final AlertDialog.Builder builder1 = new
AlertDialog.Builder(LogIn.this);
        progressBar.setVisibility(View.VISIBLE);
        login.setVisibility(View.GONE);

        if (isOnline()){

            StringRequest stringRequest = new
StringRequest(Request.Method.POST, Url, new Response.Listener<String>() {
                @Override
                public void onResponse(String response) {
                    try {
                        JSONObject jsonObject = new JSONObject(response);
                        String code = jsonObject.getString("Code");
                        String message = jsonObject.getString("Message");

                        if (code.equals("1")){
                            String fname =
jsonObject.getString("firstName");
                            String lname =
jsonObject.getString("lastName");
                            String name = fname + " " + lname;
                            String email = jsonObject.getString("email");
                            String ownerID =
jsonObject.getString("ownerID");
                            String phoneNo =
jsonObject.getString("phoneNo");
                            String vehicleID =
jsonObject.getString("vehicleID");
                            String make = jsonObject.getString("make");
                            String model = jsonObject.getString("model");
                            String year = jsonObject.getString("year");
                            String type = jsonObject.getString("type");
                            String plateNumber =
jsonObject.getString("plateNumber");
                            String color = jsonObject.getString("color");
                            String trackingDeviceID =
jsonObject.getString("trackingDeviceID");
                            String IMEI = jsonObject.getString("IMEI");
                            String IMSI = jsonObject.getString("IMSI");

                            sessionManager.createSession(name, email,
ownerID, phoneNo, vehicleID, make, model, year, type, plateNumber, color,
trackingDeviceID, IMEI, IMSI);
                            startActivity(new Intent(LogIn.this,
LocationMap.class));
                            finish();
                        }
                        else if (code.equals("2")){
                            builder1.setIcon(R.drawable.ic_error);
                            builder1.setTitle(message);
                            builder1.setMessage("Try again");
                            builder1.setPositiveButton("Cancel", new
DialogInterface.OnClickListener() {
                                @Override
                                public void onClick(DialogInterface
dialog, int which) {
                                    dialog.cancel();
```

APPENDIX G: SAMPLE PHP CODE.

StoreLocation.php

```php
<?php

$servername = "localhost";
$username = "id16231071_yobra";
$password = "SL8<!2v>vR5Z{Iof";
$database = "id16231071_trackit";

$trackingunitid = $_GET['trackingUnit'];
$latitude = $_GET['latitude'];
$longitude = $_GET['longitude'];
$speed = $_GET['speed'];

// Create connection
$conn = new mysqli($servername, $username, $password, $database);

// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}
//Perform insertion to the database
$sql = "INSERT INTO location_data (trackingUnit_id, latitude, longitude, speed
) VALUES ($trackingunitid, $latitude, $longitude, $speed);";
if($conn->query($sql) === TRUE) {
    echo "New record added successfully";
}else{
    echo "Error: " .$sql . "<br>" . $conn->error;
}

$conn->close();
?>
```

PHP code to store location data fetched by the device.

Php code to get the current location of the car.

```php
<?php

//File to connect to my database
require_once 'connect.php';

$trackingUnitID = $_POST['trackID'];

$sql = "SELECT * FROM location_data WHERE time =(SELECT MAX(time) FROM locatio
n_data WHERE trackingUnit_id = $trackingUnitID);";

//Array to store returned result
$json_array = array();

//Perform the query
if ($res = mysqli_query($conn, $sql)) {
    //If more than one row is returned by the query
    if (mysqli_num_rows($res) > 0) {
        //Fetch the rows returned
       while ($row = mysqli_fetch_assoc($res)) {
           //Insert the data values in the array
            $json_array["latitude"] = $row["latitude"];
            $json_array["longitude"] = $row["longitude"];
            $json_array["speed"] = $row["speed"];
            $json_array["time"] = $row["time"];
            $json_array["Code"] = 1;
            $json_array["Message"] = "Location data updated";

            //Encode the array in json
            echo json_encode($json_array);
       }
    }
    else {  //If there is no data for the tracking device
        $json_array["Code"] = 2;
        $json_array["Message"] = "No location data for this tracking device";
        echo json_encode($json_array);
    }
}
else {
    $json_array["Code"] = 3;
    $json_array["Message"] = "Could not perform query, SQL error";
    echo json_encode($json_array);
}
//Close the connection to the database
mysqli_close($conn);
?>
```

Php code to get the location history.

```php
<?php

//File to connect to our database
require_once 'connect.php';

//Parameters sent from the post request
$trackingUnitID = $_POST['trackID'];
$from = $_POST['from'];
$to = $_POST['to'];

//SQL query
$sql = "SELECT * FROM location_data WHERE time BETWEEN '$from' AND '$to' AND t
rackingUnit_id = $trackingUnitID;";

//Array to hold the result fetched
$json_array = array();

//Fetch rows returned by the query and store them in the array
if ($res = mysqli_query($conn, $sql)) {
    if (mysqli_num_rows($res) > 0) {
        while ($row = mysqli_fetch_assoc($res)) {
            $json_array["latitude"] = $row["latitude"];
            $json_array["longitude"] = $row["longitude"];
            $json_array["speed"] = $row["speed"];
            $json_array["time"] = $row["time"];
            $result["Code"] = 1;
            $result["Message"] = "Location data fetched";
            $result["result"][] = $json_array;
       }
       //Encode the data in json format
       echo json_encode($result);
    }
    else {
        $json_array["Code"] = 2;
        $json_array["Message"] = "No location data for this period";
        echo json_encode($json_array);
    }
}
else {
    $json_array["Code"] = 3;
    $json_array["Message"] = "Could not perform query, SQL error";
    echo json_encode($json_array);
}
mysqli_close($conn);
?>
```