



**MURANG'A UNIVERSITY OF TECHNOLOGY**

**MOBILE MECHANIC FINDER APPLICATION**

**ETABO KELVIN ESEKON**

**SC212/0743/2015**

**A PROJECT REPORT SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE IN  
PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE  
BACHELOR OF SCIENCE IN SOFTWARE ENGINEERING OF MURANG'A UNIVERSITY  
OF TECHNOLOGY**

**APRIL, 2019.**

## **DECLARATION**

I **ETABO KELVIN ESEKON**, declare that the work contained in this project is my own original work and have not been previously submitted for obtaining any other qualification.

**ETABO KELVIN ESEKON**

**DATE:** .....

**SC212/0743/2015**

**SIGN:** .....

This is to certify that this project has been submitted for examination with my approval as the supervisor.

**Mr. Ndia John**

**DATE:** .....

**SIGN:** .....

## **DEDICATION**

This project is dedicated to my father, who taught me that the best kind of knowledge to have is that which is learned for its own sake. It is also dedicated to my mother, who taught me that even the largest task can be accomplished if it is done one step at a time and not forgetting Dr. Joseph Esekun who was always there to give me piece of advice and mentorship towards my life. All thanks to you.

## **ACKNOWLEDGEMENT**

Mr. **NDIA** John has been the ideal project supervisor. His sage advice, insightful criticisms, and patient encouragement aided the writing of this project in many ways. I would also like to thank my fellow classmates whose steadfast support of this project was greatly needed and deeply appreciated. Thank you all.

## **ABSTRACT**

This project is heavily based on the mechanic finder application where the car owners and drivers will be certain to find a mechanic who will help them in a critical situation along the highways. questionnaire has been used in this project to collect the necessary information needed from the user concerning the problem they are facing and the Android studio and other relevant tools such as firebase which is real time database has been employed to develop this application.

## Table of Contents

DECLARATION .....	i
DEDICATION .....	ii
ACKWOLEDGEMENT .....	iii
ABSTRACT.....	iv
CHAPTER ONE: INTRODUCTION .....	1
1.1 BACKGROUND OF THE STUDY.....	1
1.2 PROBLEM STATEMENT;.....	1
1.3 OBJECTIVES .....	2
1.3.1 General objectives.....	2
1.3.2 specific objectives.....	2
1.4 SIGNIFICANT OF THE STUDY.....	2
1.5 SCOPE .....	2
CHAPTER TWO: LITERATURE REVIEW .....	3
2.1: REQUIREMENT GATHERING.....	3
2.1.1 INTERVIEWS .....	3
2.1.1.1 ADVANTAGES .....	3
2.1.1.2 DISADVANTAGES;.....	3
2.1.2: SURVEY .....	4
2.1.2.2.3 CONCLUSION ON REQUIREMENT GATHERING TECHNIQUES .....	4
2.2: DESIGN AND IMPLEMENTATION.....	4
2.2.1 INCREMENTAL OR ITERATIVE.....	4
2.2.2 SPIRAL MODEL.....	5
2.2.3 PHASING .....	5
2.2.4 DIRECT .....	5
2.3.1: THE MOTOR OMBUDSMAN; .....	6
2.3.2: STRENGTH OF THE MOTOR OMBUDSMAN .....	6
2.3.3: WEAKNESS OF THE MOTOR OMBUDSMAN .....	6
2.4.1: AUTOCURADOR (MANGALURU) .....	6
2.4.2 STRENGTH OF THE AUTOCURADOR .....	7
2.4.3: WEAKNESS OF THE AUTOCURADOR .....	7
CHAPTER THREE: RESEARCH METHODOLOGY .....	8
3.1 INTRODUCTION .....	8

3.2 METHODS OF RESEARCH .....	8
3.3 SAMPLING TECHNIQUES .....	8
3.4 RESEARCH INSTRUMENTS;.....	8
3.5 PROCEDURES OF DATA GATHERING .....	9
3.6 SOFTWARE DEVELOPMENT METHODOLOGY .....	9
3.6.1 ITERATIVE.....	9
3.6.2 ITERATIVE MODEL DESIGN .....	9
3.7 TOOLS.....	10
CHAPTER FOUR: SYSTEM ANALYSIS .....	11
4.1 INTRODUCTION .....	11
4.1.1 CONTEXT DIAGRAM.....	11
4.1.2 USE CASE.....	11
4.1.3 EVENTS LIST.....	12
4.1.4 INPUTS.....	12
4.1.5 OUTPUTS.....	13
4.1.6 PERFORMANCE REQUIREMENT .....	13
4.1.7 SECURITY MEASURES.....	13
4.1.8 SYSTEM REQUIREMENT .....	13
CHAPTER FIVE: SYSTEM DESIGN .....	14
5.1 DFD (data flow diagram) .....	14
5.1.1 SYSTEM DESIGN ERD .....	15
5.1.2 Driver Registration.....	15
5.1.3 Driver Login.....	17
5.1.4 Mechanic Registration .....	18
5.1.5 Mechanic Login .....	20
5.2 DATABASE DESIGN .....	22
5.2.1 ERD DATABASE DESIGN.....	23
CHAPTER SIX: CODING AND TESTING .....	25
6.2 TESTING.....	38
CHAPTER SEVEN: CONCLUSION AND RECOMMENDATION .....	41
7.1 CONCLUSION.....	41
7.2 RECOMMENDATION .....	41
REFERENCES.....	42

APPENDICES .....	43
APPENDIX I: USER MANUAL.....	43
APPENDIX 2: INSTALLATION GUIDE. ....	45
APPENDIX 3: QUESTIONNAIRE .....	46



## CHAPTER ONE: INTRODUCTION

### 1.1 BACKGROUND OF THE STUDY.

Mobile mechanic finder is a mobile application that allows car owners and the motorist to locate the nearest mechanic by calling the mechanic. Some of the widely most known applications are; The motor ombudsman (UK), Autocurador (India), Gumtree (south Africa) and AA Kenya (automation association of Kenya).

The **motor ombudsman** is a web-based auto mechanic application that was initiated in the United Kingdom to help the car owners to get car services. It's offers services such as car servicing, car repairs, tyres, exhaust and more. For a basic garage search a user just enters the town or postcode and click search. (Ombudsman, 2016)

**Autocurador (mangaluru)** application was designed by students of NIT.K ((National Institute of Technology Karnataka), India. Autocurador (Mangaluru) offers a wide range of services, right from engine servicing and wash, through a network of service stations in that particular locality where the customer wants to book the service from and it also offers breakdown services. (Chokra, 2017)

**AA KENYA** is an online auto mechanic web based in Kenya that plays a big role in helping the motorist who are frustrated and desperate to get car services along the road. It Offers services such as maintenance and repair of vehicles, road mapping and the setting up of petrol depots, all of this service requires the customer to log into AA Kenya websites for them to get assigned a technical mechanic depending on the kind of the problems they have. (Fenzi, 1971)

The problem with these web-based auto mechanic applications are, they consume a lot of data when laptops and computers are used, and memorizing the universal resource locator (URL) of the page is also a big issue. Personal computers are cumbersome even to carry while on transit. These issues actually make the existing applications unreliable and inefficient to rely on.

Therefore, mechanic finder application has made it possible for every individual to access and locate the nearest available mechanic by just logging to their apps on their phones and call mechanic via the internet.

### 1.2 PROBLEM STATEMENT;

The issue of cars breaking down along the highways is very unpredictable. Therefore, Kenyan highways is no exception to this. Long-distance drivers and any other persons on transit are in great danger when their trucks and cars breakdown. They will be stuck there for long time before being rescued or attended to.

The study done in United Kingdom by ombudsman, was trying to mitigate this problem. Ombudsman came up with the motor ombudsman, a web-based application that could help eliminate this problem once and for all, the user was expected to key in the town postcode to locate the nearest available cars service station. Students from NIT.K (National Institute of Technology Karnataka) in India formed a web-based application called AUTOCURADOR, they also seek to solve the same problem. Other examples are GUMTREE in south Africa and AA Kenya (Automation Association of Kenya).

However, the car owners and the motorists were still experiencing problems despite the development of these applications, the applications put in place are not portable and accessible by large number of people because they require users to have laptops and computer with them which. This is quite discouraging, disturbing, annoying and cumbersome to even carry it on transit, the device mentioned above requires strong WIFI or internet connection, consumes a lot of data and they are expensive even to afford. Therefore, due to intermittent internet connection in some places this approach did not operate very well.

In response to this problem, Mechanic finder application has managed to change the platform from being the web-based to mobile based application to make it more convenient, efficient and reliable to the car owners and motorist in order to get and locate the mechanic. Mechanic finder enable the user to locate the nearest mechanic possible and gives the driver the phone number and the full names of that mechanic likewise to the mechanic, which will make it possible for the driver to call the mechanic whenever necessary.

### **1.3 OBJECTIVES**

#### **1.3.1 General objectives**

To develop mobile mechanic finder application.

#### **1.3.2 specific objectives**

- i. To gather requirements that will help in the development of mechanic finder app.
- ii. To design mechanic finder app.
- iii. To develop an application that is efficient and reliable to the customers.
- iv. To implement and validate an application

### **1.4 SIGNIFICANT OF THE STUDY**

This application is expected to be of great benefits to the following;

**Car owner's and the motorist.** The system has actually provided real time communication and feedback to the stranded motorist and car owners who are in need of a mechanic. This application has helped them to locate the nearest mechanic.

**Country has a nation;** A mechanic finder application has created job opportunities to large population of youth who have skills in mechanic because mechanic personnel are required in large number to help the entire population

### **1.5 SCOPE**

This system has been limited within the country (Kenya). Mechanic finder application has mitigated the problem affecting all riders all over the nation. This has made the transport system in the country to run smoothly and operational.

## CHAPTER TWO: LITERATURE REVIEW

### 2.1: REQUIREMENT GATHERING.

Requirement gathering is a procedure used to collect information about a specific or a particular problem in a particular domain. Therefore, here are some of the techniques used to collect data. These includes;

#### 2.1.1 INTERVIEWS

An interview is a conversation where questions are asked and answers are given. In common parlance, the word "interview" refers to a one-on-one conversation between an interviewer and an interviewee. Therefore, requirement gathering can be conducted using this technique because it actually involves the end user and the developer. The end user can be able to tell where the problem is in the existing systems and the developer can try every possible way to resolved and improve on the same problem stated by the end user. (Johnson, L. 2013).

##### 2.1.1.1 ADVANTAGES

**Accurate screening,** face to face interview helps with more accurate screening. The individual being interviewed is unable to provide false information in most occasions. Therefore, the researcher is as good as collecting the right and correct data of information concern a specific problem in a specific domain.

**Capture emotion and behaviour,** there is no doubt in interviews because the researcher can possibly study the behaviour and the emotion of the person being interviewed. Therefore, the researcher can definitely know if the collected data is correct or not

**Keep focus,** the interviewer is the one having control over the interview and can keep the interviewee focused and on track to completion. This gives an advantage to the one conducting a study on the existing problem because he/she will be getting real experience of the problem in that domain from the horse mouth.

##### 2.1.1.2 DISADVANTAGES;

**Cost is a major disadvantage for face-to-face interviews.** They require a staff of people to conduct the interviews, which means there will be personnel costs. Personnel are the highest cost a business can incur. It's difficult to keep costs low when personnel are needed.

**The quality of data** you receive will often depend on the ability of the interviewer. Some people have the natural ability to conduct an interview and gather data well. The likelihood of the entire interviewing staff having those skills is low. Some interviewers may also have their own biases that could impact the way they input responses.

**Manual data entry,** If the interview is administered on paper, the data collected will need to be entered manually, or scanned, Data entry and scanning of paper questionnaires can significantly increase the cost of the project. A staff of data entry personnel will need to be hired. Mobile surveys on iPads, tablets, or other mobile devices can cut-down on manual data entry costs and information is ready for analysis.

### 2.1.2: SURVEY

A survey is a data gathering method that is utilized to collect, analyse and interpret the views of a group of people from a target population. Surveys have been used in various fields of research, such as sociology, marketing research, politics, psychology etc. this technique is the most of the convenient technique to collect a lot of data and it is cost effective. (Schwarz, 2016)

#### 2.1.2.1 ADVANTAGES

**Low cost:** When conducting surveys, you only need to pay for the production of survey questionnaires. On the other hand, other data gathering methods such as focus groups and personal interviews require researchers to pay more.

**Convenient data gathering,** the online survey method has been the most popular way of gathering data from target participants. Aside from the convenience of data gathering, researchers are able to collect data from people around the globe.

**High representativeness,** Surveys provide a high level of general capability in representing a large population.

**Precise results,** they provide uniform definitions to all the subjects who are to answer the questionnaires. Thus, there is a greater precision in terms of measuring the data gathered.

#### 2.1.2.2: DISADVANTAGES

**Inflexible design,** the survey that was used by the researcher from the very beginning, as well as the method of administering it, cannot be changed all throughout the process of data gathering.

**Not ideal for controversial issues,** Questions that bear controversies may not be precisely answered by the participants because of the probably difficulty of recalling the information related to them.

#### 2.1.2.2.3 CONCLUSION ON REQUIREMENT GATHERING TECHNIQUES

In the Motor Ombudsman and Autocurador application requirement gathering was done using interviews.

In this project, Mechanic finder has considered taking a survey as a convenient technique for requirement gathering. Survey is convenient in the sense that you can gather a lot of information within the shortest time possible. The cost to conduct survey is cheaper than conducting the interview to gather information.

## 2.2: DESIGN AND IMPLEMENTATION

A design is simply the creation of a plan or convention for the construction of the object, system and measurable human interaction.

Implementation method is a systematically structured approach to effectively integrate a software-based service or component into the workflow of an organizational structure or an individual end-user. System design and implementation techniques are as follows;

### 2.2.1 INCREMENTAL OR ITERATIVE

Incremental approach is dividing the project in various independent parts and developing these sub-parts at the same rate or different rate and integrating them when ready. These can be completed and integrated into a common repository as they become ready. Once these parts are ready, next set is picked. It is also possible that all the parts can be simultaneously worked on and integrating them when ready in the central repository.

In this technique the system is developed in phases to make sure that the errors are minimal or zero before the full system is developed. For example, you can start by designing the interface and then the backend part of the system separately then later you integrate them together to form a full system. (Wagner, R. 2006, October).

This technique is more important because it is easier to debug and test the system during a smaller integration, more flexible in the sense that it is less costly to change scope and requirement, each iteration is easily managed milestone and it is easier to manage risk because risk pieces are identified and handled during its iteration

The technique had a number of disadvantages for example each phase is rigid and do not overlap each other and problems may arise pertaining to systems architecture because not all requirements are gathered up front for entire system.

### 2.2.2 SPIRAL MODEL

Spiral model is a combination of waterfall and iterative model, each phase in spiral model begins with the design goals and end with the client receiving the progress. Spiral SDLC model starts with a small set of requirements and goes through each development's phases for those sets of requirements. (Boehm, B.2011).

The spiral life cycle is shown as a spiral model that begins with the planning phase first from the centre (inward) of the spiral, eventually working its way outward, over and over again, until completion of the project. The planning phase will include activities such as feasibility study, a survey of user's requirements, overall design choice, generation of implementation alternative, and implementation strategy. The purpose of this phase is to have enough information to build a prototype.

Spiral model is important in the sense it is fast and its features are added in a systematic way to the system and there is always a space for customer feedback.

Looking for its disadvantages it actually works best for large projects, documentation is more as it has intermediate phases and its smooth operation needs to be followed strictly, this means that if you don't follow its rules and procedures you may end up building a wrong thing altogether.

### 2.2.3 PHASING

It is the method of system implementation that involves changing from existing system to a new one that takes place in stages. The Autocurator used this method to implement their system. The importance of this method is that the issues of around scale can be addressed without major impact and the training can be completed in small parts, but there is also a big problem related to this technique, the problem is that it takes a lot of time to get the system fully online than other methods and there is a possibility of data loss if part of the system fails. (Breton, G. 2014, August).

### 2.2.4 DIRECT

This is the implementation method used to implement the system when no phased or pilot is needed. It actually makes sure that the old system goes off or retired and the new system to be developed goes live. The method has been used in the development of the motor ombudsman and the Autocurator application.

The importance of this technique is that it does not need the system to be more critical. The problem is that if you are not sure how the system will work. (Govokhina,2014)

#### 2.2.4.3 CONCLUSION ON DESIGN AND IMPLEMENTATION TECHNIQUES

In the design and implementation of this system, iterative method has been used to design the system because it is very easy to monitor the progress of the system given that every phase is developed iteratively then later integrated to a full system and the phasing method to implement the system simply because in phasing you can deal with issues around scale with no impact and the issue of correctness is observed because there will be minimal error made.

#### 2.3.1: THE MOTOR OMBUDSMAN;

The motor Ombudsman is a web-based auto mechanic that was initiated in the Westminster, London, it is the first voluntary and fully-impartial private sector ombudsman that provides a self-regulatory environment for automotive industry using its chartered trading standards institute approved motor industry code of practices.

Garage finders covers the whole of the UK and Northern Ireland making it more suitable for the civilian to locate the garage and their problems resolved within the shortest time possible. The user was expected to key in the town codes and locate garage near to them.

The **Motor Ombudsman** offers services such as car servicing, car repairs, tyres, exhaust and more. For a basic garage search a user just enters the town or postcode and click search. (Ombudsman, 2016)

#### 2.3.2: STRENGTH OF THE MOTOR OMBUDSMAN

The motor ombudsman is a very powerful mechanic and garage finder platform in the United Kingdom. This platform provides real time communication and feedback between the users or the car owners and the mechanic, it also provides varieties of car service therefore the consumer can minimise time wastage for locating other services and cost-efficient. (Ombudsman, 2016)

#### 2.3.3: WEAKNESS OF THE MOTOR OMBUDSMAN.

The motor ombudsman has weakness just like any other platforms in the globe. Computers and laptops are required to access car services stations. These devices are expensive and consumes a lot of data.

Secondly ombudsman powers are completely inconsistent with varying standards for membership, authority and the ability to enforce decisions. This “devalues” the meaning of the term ombudsman.

Thirdly uploading document on communication ombudsman sites not the easiest. (Ombudsman, 2016)

#### 2.4.1: AUTOCURADOR (MANGALURU)

This application was designed by students of NIT.K(National Institute of Technology Karnataka), India. The founders Godana Dili and Kumar Gunda when asked about the idea for starting up the application, they said “personal experienced that happen a year ago was the reason we had gone on a long drive when a bike parked on the road fell on our vehicle and resulted in a dent on our SUV”. when they approached the nearest car showroom, they quoted a hefty price to clear the dent. Through this life experienced the dual just decide to develop a web-based application to help other population in India from such worst scenarios.

Autocurador (Mangaluru) offers a wide range of services, right from engine servicing and wash through a network of service stations in that particular locality where the customer wants to book the service from

and it does offer breakdown services and the best part is that it also offers pick and drop facility. (Chokra, 2017)

#### 2.4.2 STRENGTH OF THE AUTOCURADOR

This application brought a good experienced to the Indian people at large, the car owners were able to book a service at any stations near their locality rather than hiring more expensive car showrooms at the eve of the car breakdown. what was needed was just to sign up or sign in for the app to locate the nearest available services stations.

Another major advantage was the convenience factor, mechanic was even able to work on your vehicle while you are at work.

#### 2.4.3: WEAKNESS OF THE AUTOCURADOR

This application requires strong internet connection and the devices used to connect to the mechanics are expensive to buy and consumes a lot of data.

#### 2.4.4 CONCLUSION

In the previous studies conducted to solve the existing problem, majority of the developed web-based application tried to make sure that the car users are able to get cars service stations at ease. The motor Ombudsman, the Autocurador and AA Kenya were few auto mechanic applications put in place to locate car service stations. These applications have really helped a lot, in the sense that booking a car service at your place of work and at home has become very easy and less costly.

From previous studies conducted, car owners were expected to have sophisticated devices such as laptops and computers to access car services station and others were compelled to go to cyber cafes to do the same process. Laptops are expensive and cumbersome to carry on transit.

Mechanic finder application has helped users to get access to mechanic services by just calling a mechanic. Mobile phones are portable, less expensive and convenient as compared to laptops and desktops computers.

## **CHAPTER THREE: RESEARCH METHODOLOGY**

### **3.1 INTRODUCTION**

This chapter present the discussion on the project methodology, the subjects, sampling techniques, projects instruments, procedure of data gathering and statistical treatment that will be used for accurate data analysis and interpretation.

### **3.2 METHODS OF RESEARCH**

The descriptive, survey and experimental methods of research have been used in this project. Descriptive because its main concern is to get the view of the characteristic of the subject exposed to the motorist when their car breakdown on roads as well as those on board.

This study was also experimental because we compared between the previously developed web base application and Mechanic finder in relation to their performance. Both qualitative and quantitative methods have been observed since they elicit opinions and numerical data from the respondent through survey and questionnaire.

### **3.3 SAMPLING TECHNIQUES**

Convenience sampling has been utilized in this project. Clustered sampling will be used in this project because the entire population was divided into subgroups which were randomly selected to be included in the study. Cluster is usually already defined. Cluster sampling is more efficient than simple random sampling, especially where a study took place over a wide geographical region, this is because it is easier to contact a lot of individuals in a few GP practices than individual in many different GP practices. (Cochran, 1977)

### **3.4 RESEARCH INSTRUMENTS;**

The project has used only two research tools to obtain data from the respondent.

#### **3.4.1 SURVEY**

A survey instrument is a tool for consistently implementing a scientific protocol for obtaining data from respondents, the instruments will involve a questionnaire that provides a script for presenting a standard set of questions and responses options to the respondents. This survey has involved questions that address a specific study and what may be done to resolve the problem and the questionnaire responses should be augmented by other kinds of measurements derived from the instrument.

#### **3.4.2 APPLIED**

Applied research instrument involves questionnaire, interviews and observation. This tool has been maximized in this project to collect a curate data from the population. From this study apart from survey, interviews and the observation, questionnaire have been imposed to collect data from the respondent.



### **3.5 PROCEDURES OF DATA GATHERING**

Data gathering is a procedure of collecting data from the participants, this is the most vital and important part of any study carried out to solve a specific problem. The study has used survey since the study is trying to address the issue that is affecting the entire nation and taking interviews is expensive. Survey involves development of questionnaires, observation and focus groups. The study has actually set on questionnaire. The study has employed the use of questionnaire because it is efficient and less time consuming and it is easy to collect data across the entire population of the study. (Barber, 2008)

### **3.6 SOFTWARE DEVELOPMENT METHODOLOGY**

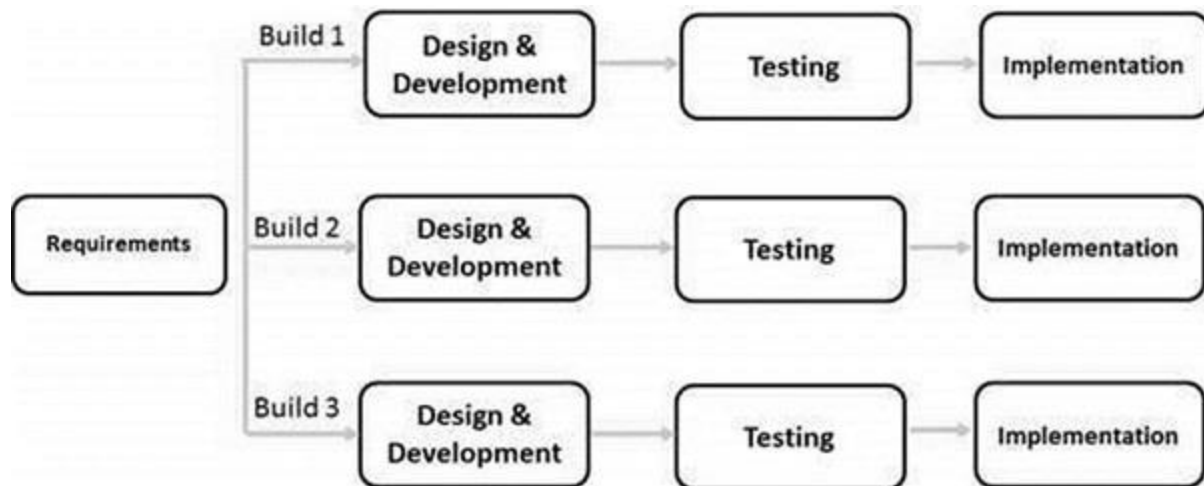
Software development process is the process of dividing software development work into distinct phases to improve design, product, management and project management. The methodology may include the pre-definition of the specific deliverables and artefacts that are created and completed by project team to develop and maintain an application.

#### **3.6.1 ITERATIVE**

Mechanic finder application has employed the use of iterative. The basic idea behind this method is, it develops a system through a repeated cycle and in smaller portions at a time, allowing software developer to take advantage of what was learned during development of earlier parts of the system. Learning came from both development and use of the system, where possible key steps in the process start with simple implementation of subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented. (Tarhini, 2018)

#### **3.6.2 ITERATIVE MODEL DESIGN**

Iterative and incremental development starts with a simple implementation of the subsets of the software requirement and iteratively enhances the evolving versions until the full system is implemented. at each iteration, design modifications are made and new functional capabilities are added.



Iterative and Incremental development is a combination of both iterative design or iterative method and incremental build model for development. "During software development, more than one iteration of the software development cycle may be in progress at the same time." This process may be described as an "evolutionary acquisition" or "incremental build" approach." (Zykov, 2018)

In this incremental model, the whole requirement was divided into various builds. During each iteration, the development module goes through the requirements, design, implementation and testing phases. Each subsequent release of the module adds function to the previous release. The process continues till the complete system is ready as per the requirement.

The key to a successful use of an iterative software development lifecycle is rigorous validation of requirements, and verification & testing of each version of the software against those requirements within each cycle of the model. As the software evolves through successive cycles, tests must be repeated and extended to verify each version of the software.

Mechanic finder application has employed this model because, it's working functionality is quickly developed and early in the life cycle, results are obtained early and periodically, parallel development is planned for example in the development of the mechanic finder application, Mechanic app and the Rider's app, progress can be measured and less costly to change the scope.

However, its disadvantage was inevitable because it needs more resources, more management attention is required and defining increments may require definition of the complete system.

### 3.7 TOOLS.

Mechanic finder has employed the use of Android studio as code editor, Firebase as a real time database to store data of users and to monitor them at ease, and Java as a programming language.

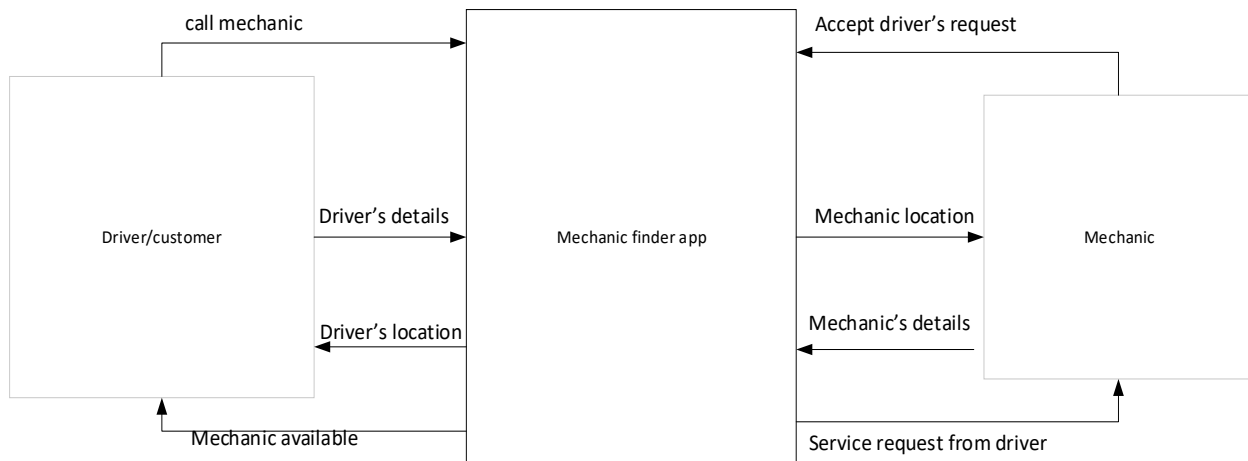
## CHAPTER FOUR: SYSTEM ANALYSIS

### 4.1 INTRODUCTION

The main objective of the system analysis phase is to understand the system being proposed and to ensure that it support the requirements. This phase includes the building of the concrete foundation of the system development. The purpose of the mechanic finder app is to make sure that all car owners and motorist gain access to quality services by enabling them to interact with qualified mechanics country wide who will be there for them at a click of the button” Call Mechanic.” This application will make a mechanic find drivers at ease.

#### 4.1.1 CONTEXT DIAGRAM

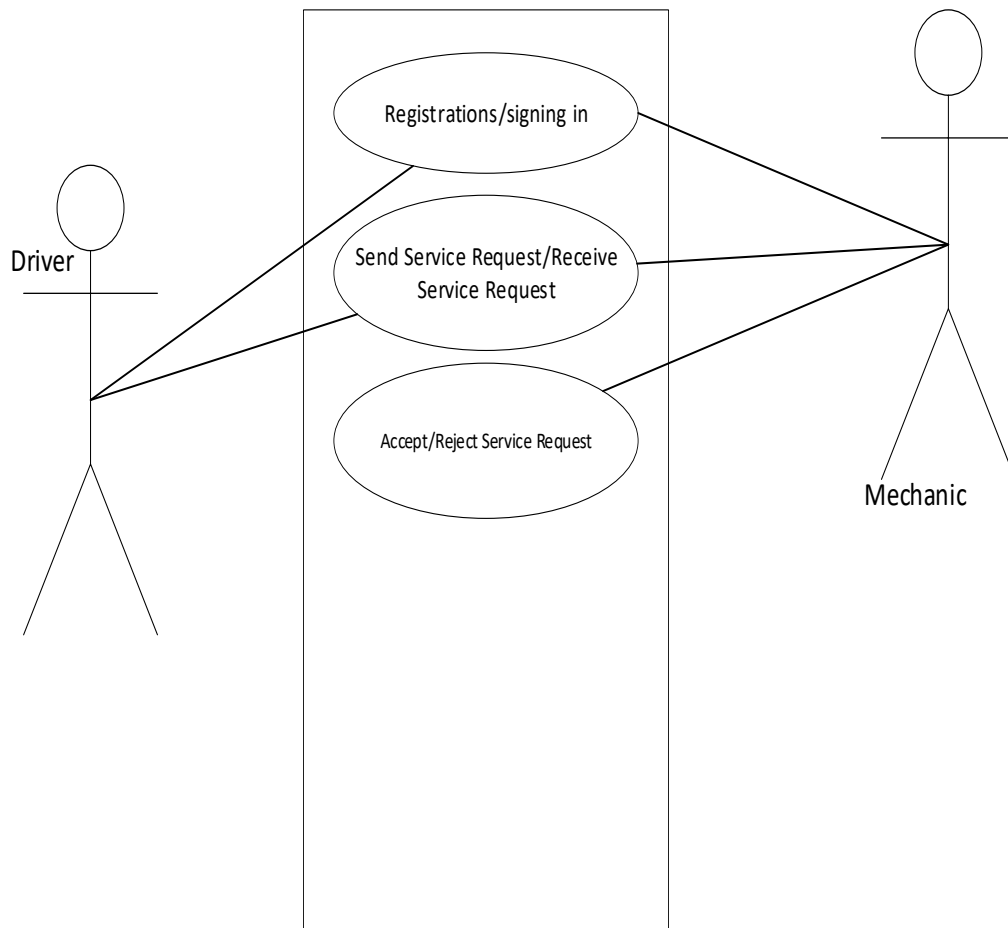
Context diagram is a diagram that defines the boundary between the system, or part of a system and its environment, showing the entities that interact with it. This diagram is a high-level view of the system. The objective of the system context diagram is to focus attention on external factors and events that should be considered in developing a complete set of systems requirements and constraints. It has a data flow connector showing how information and communication flows in the system. For example, the driver may decide to call a mechanic and the mechanic is in the position to receive the notification and consequently accept the request or reject it.



**figure 4.1 context diagram**

#### 4.1.2 USE CASE

Use case diagram are used to describe a set of actions that a system should collaborate with one or more external users of the system. In this system we have only two actors, the driver who is the customer to be attended to and the mechanic. The actors should register and logins to interact with the system. The driver is supposed to call a mechanic in order to get the services he requires.



**Figure 4.2 use case diagram**

#### 4.1.3 EVENTS LIST

Mechanic finder application provides the following events upon its operations;

1. Mechanic and the driver sign up for the application if they are not existing members.
2. Mechanic should log in, to access his/her location for Service request call from the driver.
3. Drivers also logs in to identify their location to send a Service request to the mechanic.
4. Driver will be able to call for a mechanic and the mechanic can also accept the request call.

#### 4.1.4 INPUTS

Mechanic finder application has two users these are;

1. Mechanic
2. Driver

The driver will be required to register. When registering the driver is required to enter these fields; email address and password upon registering on setting menu the driver is required to add his/her name and the phone contact and check on confirm radio button. After registering the driver will be required to sign in using the credentials he provides, most likely the email address and the password, after signing in the driver will be granted full access to whatever services mechanic finder offers. The driver will be able to view his location accurately and search for a place to go and most importantly to send Service request to the mechanic. The nearest mechanic will be shown to the customer upon a click of call a mechanic button where the app will just search for the nearest available mechanic with full information about the mechanic such as the phone number and the name.

The mechanic is required to register. When registering the mechanic is required to enter these fields, email address, password, and add his/her name and phone number on setting menu upon signing. After registering the mechanic will be required to sign in using email address and the password. Consequently, the mechanic will be taken to his maps where he will be able to receive Service request from the driver. The mechanic is given a privilege and power to accept or reject Service request from the customer. Upon the acceptance of the request, the mechanic will be showed the place he is expected to meet the customer on his map. The mechanic will be provided with the phone number and the name of the customer(driver) to call him if need be.

#### 4.1.5 OUTPUTS

The driver will be showed the nearest mechanic upon a click of calling mechanic button and if there will be no available mechanic the driver will be notified that there are no mechanics available and the application will extend the search further to make sure that the driver will get the help.

The mechanic will receive a notification requesting him/her for a Service request from the driver showing the real name and contact provided previously by the customer during registration period and the location where he can meet with the customer will be shown on the map.

#### 4.1.6 PERFORMANCE REQUIREMENT

System performance requirement refers to the system speed that will be used to process and fetch the data required by the user. Mechanic finder app will be in a position to make sure that the search for the mechanic is 24/7 hrs thus helping drivers get access to better services anywhere anytime.

#### 4.1.7 SECURITY MEASURES

Mechanic finder app will employ the use of log-on security through the use of email and password created by the users during registration period.

#### 4.1.8 SYSTEM REQUIREMENT

Mechanic finder app will require the users to install the app to their android phones in order to function well as required. The user will be required to make sure that the location is turn on to enable the app to fetch the accurate location of the user or the mechanic and the user will be required to allow the application to access the device location.

## CHAPTER FIVE: SYSTEM DESIGN

### 5.1 DFD (data flow diagram)

This project has adopted the data flow diagram to represents the flow of the data through the system designed. DFD will clearly elaborate the system and visualizes the data processing for example it will show the kind of data to input and output from the system and where the data will come from and where the data will be stored.

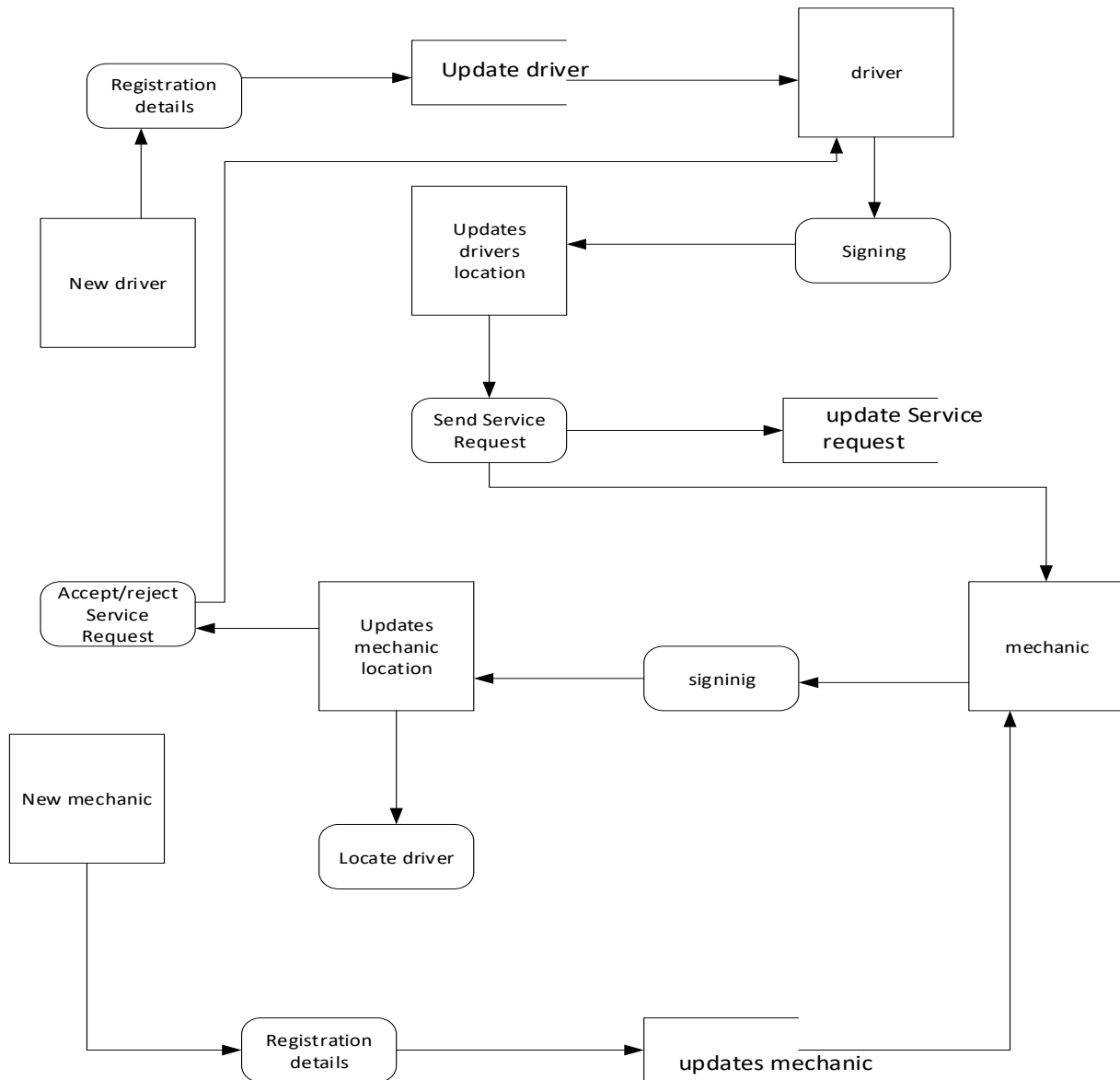
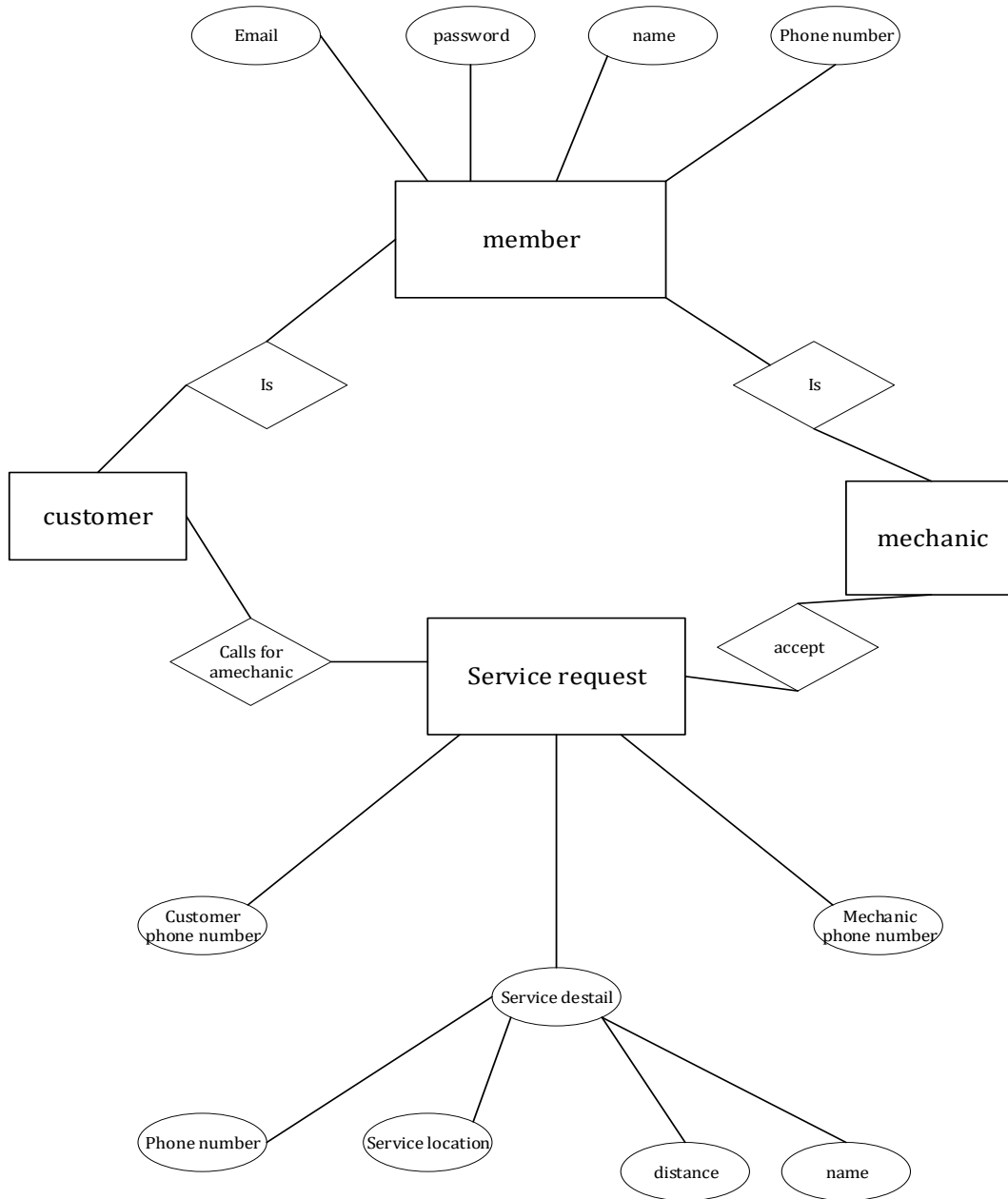


Figure 5.1 data flow diagram (DFD)

### 5.1.1 SYSTEM DESIGN ERD

Entity relationship diagram is structural diagram used to show different symbols and connectors that visualize important information. The major entities within the system scope and the inter-relationships among these entities.

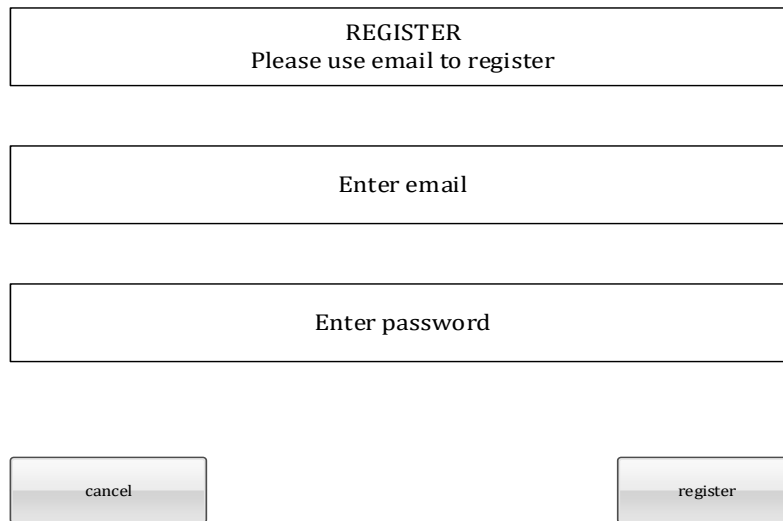


*figure 5.2 system erd (entity relationship diagram)*

### 5.1.2 Driver Registration

The driver will be required to fill in the required fields in order to be registered. During registration process the driver will be provided with a decision to make either to cancel registration or continue with registration.

Once the driver is successfully registered, he will be allowed to sign in using the credentials he provided during registration process.

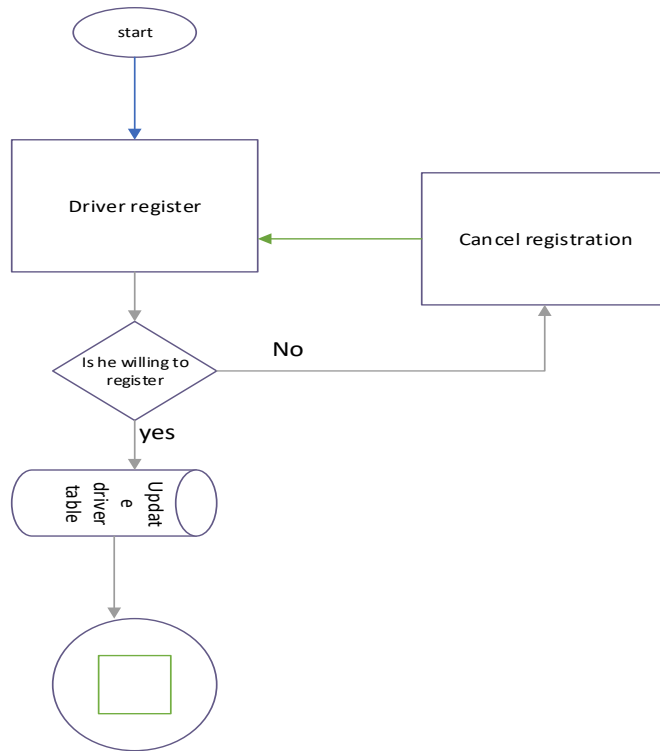


The form consists of four main components arranged vertically. At the top is a rectangular box with a black border containing the text "REGISTER" in bold, centered, with "Please use email to register" in a smaller font below it. Below this is another rectangular box with a black border containing the text "Enter email" in the center. The third component is a larger rectangular box with a black border containing the text "Enter password" in the center. At the bottom, there are two buttons: a "cancel" button on the left and a "register" button on the right. Both buttons have a light gray gradient background and a thin black border.

*Figure 5.3 driver registration*

A flow chart is a type of diagram that represents an algorithm, workflow or process. Flowchart shows step by steps the process taken by an entity in the system inform of boxes of various kind and their order by connecting the boxes with arrows. For example, here the user is expected to fill in the email and the password to register to the system. If the email and the password are correct, the system will authenticate the user to login to the system or less it prompts the user to repeat the process again





**Figure 5.3 driver's registration flow chart**

### 5.1.3 Driver Login

The driver will be required to login using the email and the password he created during the registration process. The driver will be provided with a decision to make either to cancel the login process by clicking the cancel button provided in the login interface or continue with login process by clicking on the sign in button.

**SIGN IN**  
 Please use email to sign in

Enter email

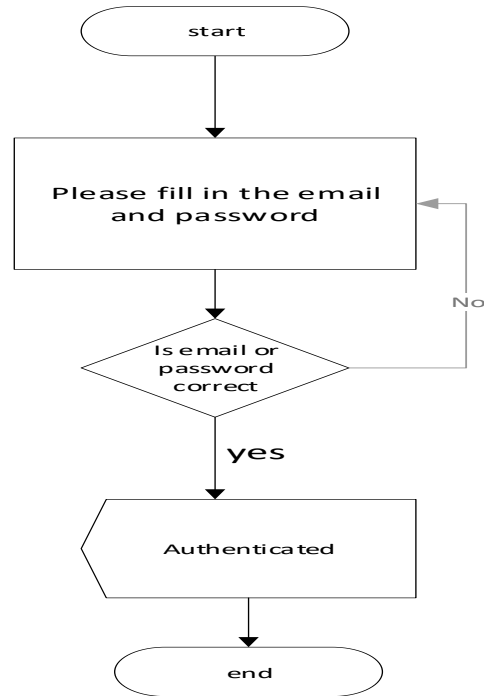
Enter password

cancel

Sign in

*Figure 5.4 driver login interface*

For example, here the user is expected to fill in the email and the password to login to the system. If the email and the password are correct, the system will authenticate the user to login to the system or less it prompts the user to repeat the process again



*Figure 5.5 driver's login flow chart*

#### 5.1.4 Mechanic Registration

The mechanic will be required to fill in the details required. During registration process the mechanic will be provided with two options either to cancel the registration or continue with registration by clicking on register button.

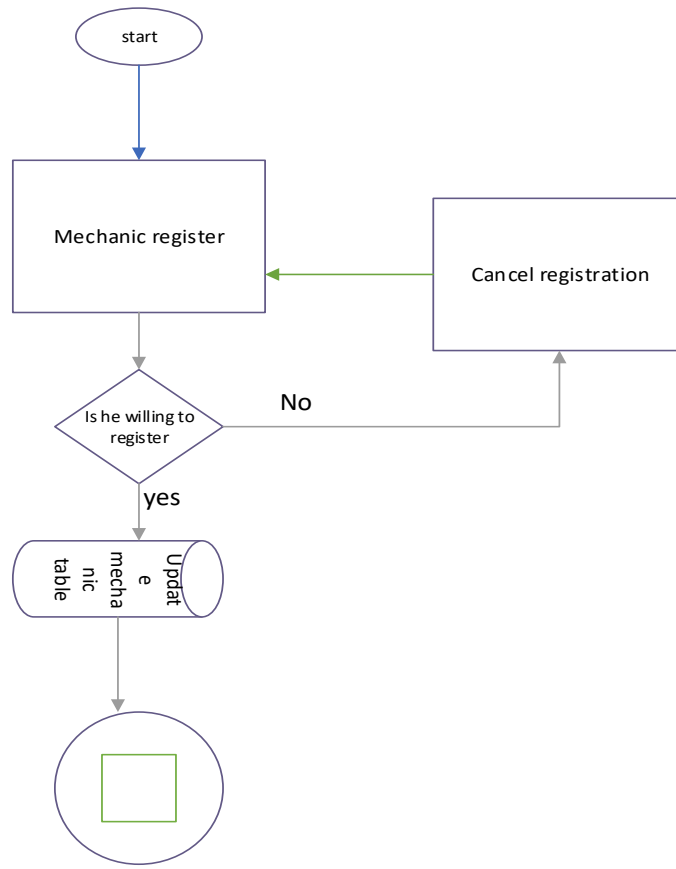


The image shows a registration form with the following elements:

- A title box containing the text "REGISTER" and "Please use email to register".
- An input field labeled "Enter email".
- An input field labeled "Enter password".
- Two buttons at the bottom: "cancel" on the left and "register" on the right.

***Figure 5.6 mechanic registration interface***

In the flowchart below, it shows how the user is expected to fill in the email and the password to register to the system. If the user is willing to register, he/she is expected to click register button to proceed. The user will be updated to the system database.



*Figure 5.7 Mechanic registration flow chart*

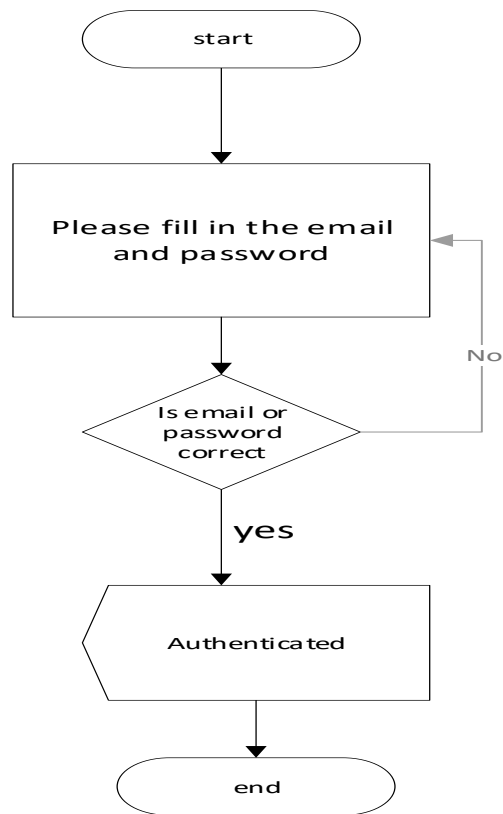
#### 5.1.5 Mechanic Login

The mechanic is required to sign in using the email and password he created during the registration process. The mechanic will be provided with the options either to cancel the sign in process or proceed to sign in by clicking the sign in button in the interface.

The diagram illustrates a login interface layout. At the top is a rectangular box containing the text "SIGN IN" in all caps, followed by "Please use email to sign in". Below this are two separate rectangular input fields, the first labeled "Enter email" and the second labeled "Enter password". At the bottom of the interface are two buttons: a "cancel" button on the left and a "Sign in" button on the right. Both buttons have a light gray gradient and rounded corners.

**Figure 5.8 Mechanic's login interface**


Here the user is expected to fill in the email and the password to login to the system. If the email and the password are correct, the system will authenticate the user to login to the system or less it prompts the user to repeat the process again.





*Figure 5.9 Mechanic login data flow chart*


## 5.2 DATABASE DESIGN


Mechanic finder application database will have the following tables which will have the various fields. The following are the database tables.

drivers	
 PK	email
password	
attribute name	
name	
phone number	
services required	

Mechanics	
 PK	email
password	
name	
phone number	
Service	

Service Request	
 PK	Customer rideID
destinationLat	
destinationLng	

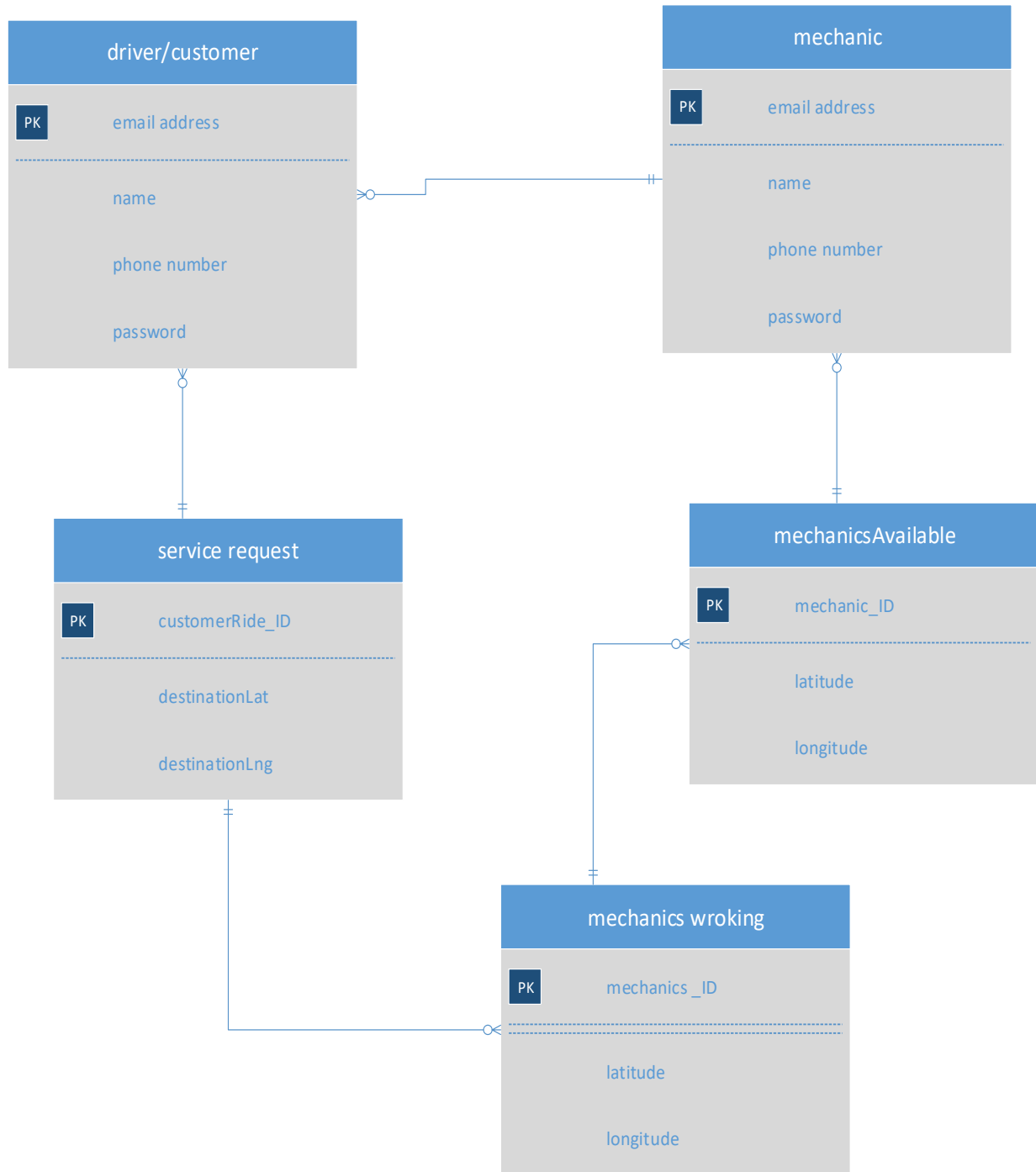
MechanicsAvailabe	
 PK	Mechanic ID
latitute	
longitude	

Mechanic working	
 PK	mechanic ID
latitude	
longitude	

*Figure 5.9 database tables*

### 5.2.1 ERD DATABASE DESIGN

Entity Relationship Diagram contains different symbols and connectors that visualize two important information: The major entities within the system scope, and the inter-relationships among these entities are shown and how they related to each other.



**Figure 6.0 ERD Database design**

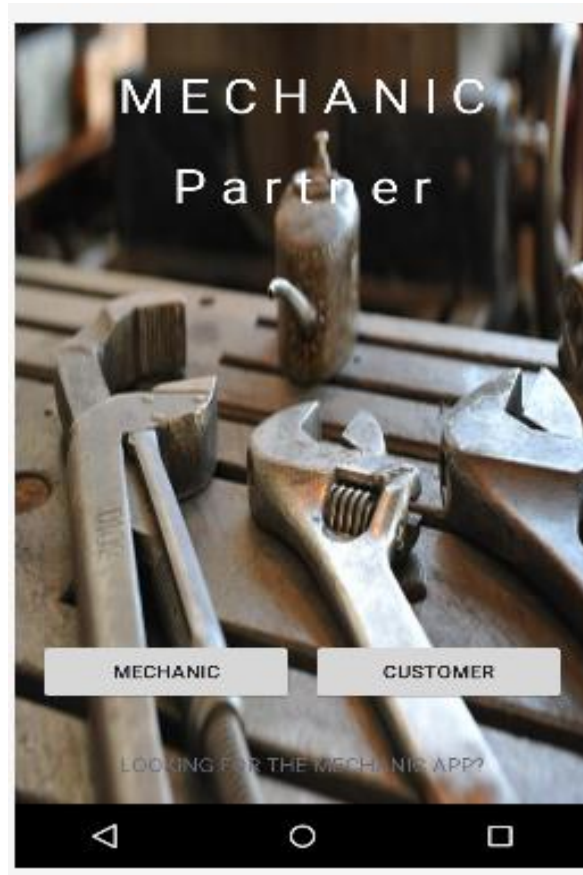


## CHAPTER SIX: CODING AND TESTING

Mechanic finder have been developed using android and java programming language. Firebase have been used as a real-time database server for this application.

The coding of the system starts with the users of the application being given privilege to choose whether they want to be a mechanic or a driver who is a customer. Mechanic has to choose a mechanic option in order to register and as well as the driver has to choose customer option to register with valid details so that they can use the same credentials to sign in to the application.

Below is the interface of the app when it launched



*Figure 6.1 app interface.*

Below are the codes for driver registration and login pages.

```

mRegistration = (Button) findViewById(R.id.registration);

mRegistration.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        final String email = mEmail.getText().toString();
        final String password = mPassword.getText().toString();
        if (TextUtils.isEmpty(email)) {
            mEmail.setError("please input email to proceed");
            mEmail.requestFocus();
        } else if (TextUtils.isEmpty(password)) {
            mPassword.setError("please input your password");
            mPassword.requestFocus();
        }

        else {
            mAuth.createUserWithEmailAndPassword(email, password).addOnCompleteListener( activity: CustomerLoginActivity
                .this, (task) -> {
                    if (!task.isSuccessful()) {
                        Toast.makeText( context: CustomerLoginActivity.this, text: "sign up error", Toast.LENGTH_SHORT).show();
                    } else {
                        String user_id = mAuth.getCurrentUser().getUid();
                        DatabaseReference current_user_db = FirebaseDatabase.getInstance()
                            .getReference().child("Users").child("Customers").child(user_id);
                        current_user_db.setValue(true);
                    }
                });
        }
    }
});

```

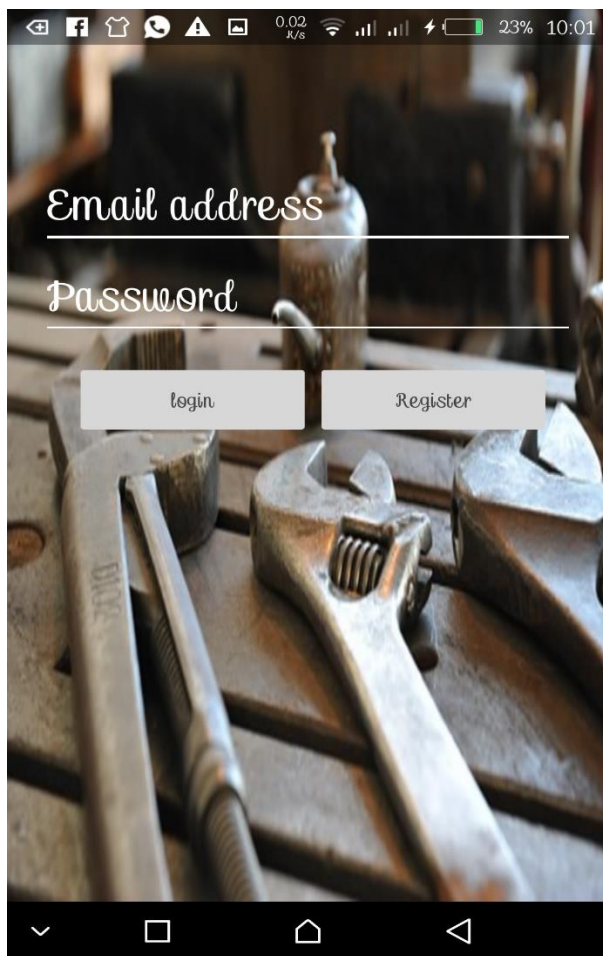
```

mLogin.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        final String email = mEmail.getText().toString();
        final String password = mPassword.getText().toString();
        if (TextUtils.isEmpty(email)) {
            mEmail.setError("please input your email");
            mEmail.requestFocus();
        } else if (TextUtils.isEmpty(password)) {
            mPassword.setError("please input your password");
            mPassword.requestFocus();
        }

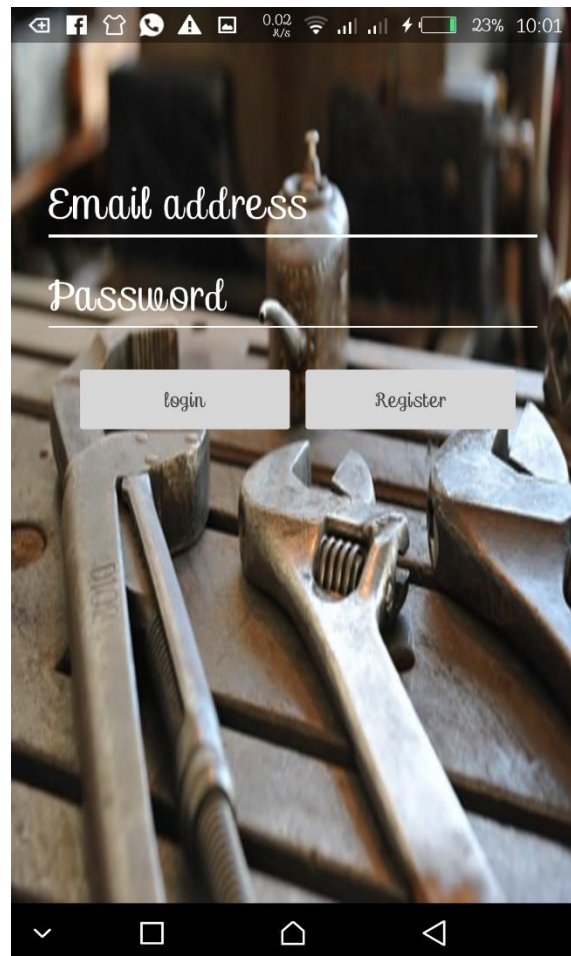
        else {
            mAuth.signInWithEmailAndPassword(email, password).addOnCompleteListener( activity: CustomerLoginActivity.
                this, (task) -> {
                    if (!task.isSuccessful()) {
                        Toast.makeText( context: CustomerLoginActivity.this, text: "sign in error", Toast.LENGTH_SHORT).show();
                    }
                });
        }
    }
});

```

Here is the interface for driver registration. And login interface.



*Figure 6.2 registration interface*



*Figure 6.3 login interface*

Below are the codes for mechanic registration page.

```

mEmail = (EditText) findViewById(R.id.email);
mPassword = (EditText) findViewById(R.id.password);

mLogin = (Button) findViewById(R.id.login);
mRegistration = (Button) findViewById(R.id.registration);

mRegistration.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        final String email = mEmail.getText().toString();
        final String password = mPassword.getText().toString();
        if (TextUtils.isEmpty(email)){
            mEmail.setError("please input your email to proceed");
            mEmail.requestFocus();
        }else if (TextUtils.isEmpty(password)){
            mPassword.setError("please input your password");
            mPassword.requestFocus();
        }
        else{
            mAuth.createUserWithEmailAndPassword(email, password).addOnCompleteListener( activity: DriverLoginActivity.this, (task)
                if(!task.isSuccessful()){
                    Toast.makeText( context: DriverLoginActivity.this, text: "sign up error", Toast.LENGTH_SHORT).show();
                }else{
                    String user_id = mAuth.getCurrentUser().getUid();
                    DatabaseReference current_user_db = FirebaseDatabase.getInstance().getReference().child("Users").child(
                        current_user_db.setValue(email);
                    }
            }
        });
    }
});

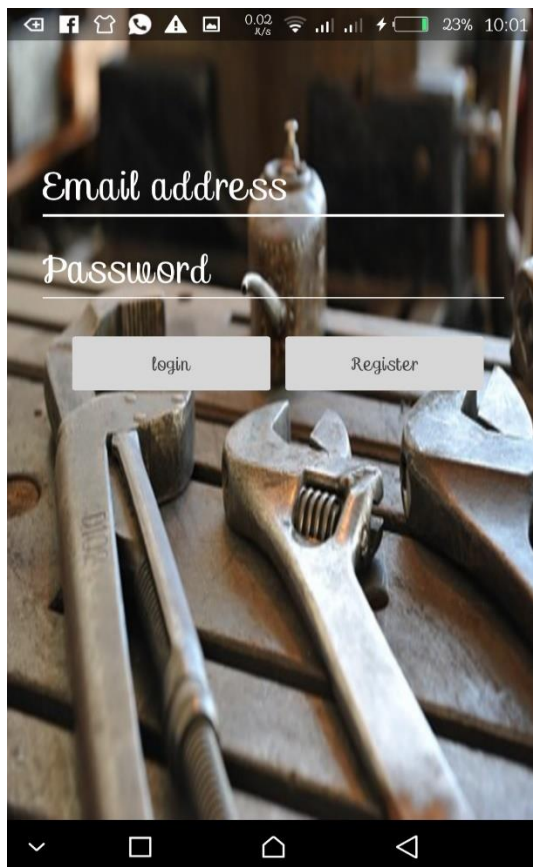
```

```

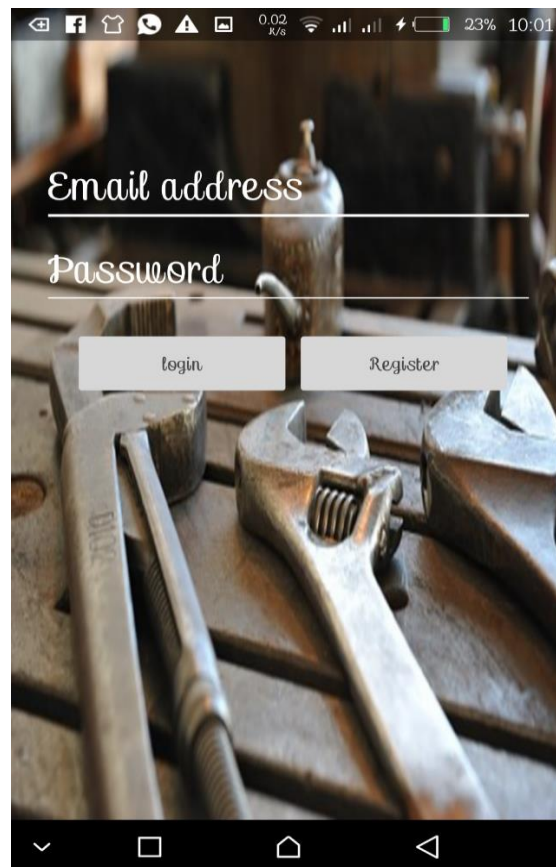
mLogin.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        final String email = mEmail.getText().toString();
        final String password = mPassword.getText().toString();
        if (TextUtils.isEmpty(email)){
            mEmail.setError("please input your email");
            mEmail.requestFocus();
        }
        else if (TextUtils.isEmpty(password)){
            mPassword.setError("please input your password");
            mPassword.requestFocus();
        }
        else{
            mAuth.signInWithEmailAndPassword(email, password).addOnCompleteListener( activity: CustomerLoginActivity.
                this, (task) -> {
                    if(!task.isSuccessful()){
                        Toast.makeText( context: CustomerLoginActivity.this, text: "sign in error", Toast.LENGTH_SHORT).show();
                    }
                });
        }
    }
});

```

Below is the interface for mechanic registration and login pages.



*Figure 6.4 registration interface*



*figure 6.5 login interface*

The following java code for driver's or customers maps and all the actions necessary for location of the mechanic.

```
package com.etabo.esekon;

import ...

public class CustomerMapActivity extends FragmentActivity implements OnMapReadyCallback {

    private GoogleMap mMap;
    Location mLastLocation;
    LocationRequest mLocationRequest;

    private FusedLocationProviderClient mFusedLocationClient;

    private Button mLogout, mRequest, mSettings, mHistory;

    private LatLng pickupLocation;

    private Boolean requestBol = false;

    private Marker pickupMarker;

    private SupportMapFragment mapFragment;

    private String destination, requestService;

    private LatLng destinationLatLng;

    private LinearLayout mDriverInfo;

    private ImageView mDriverProfileImage;

    private TextView mDriverName, mDriverPhone, mDriverCar;

    private RadioGroup mRadioGroup;
```





```
private RatingBar mRatingBar;
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_costumer_map);  
    // Obtain the SupportMapFragment and get notified when the map is ready to be used.  
  
    mFusedLocationClient = LocationServices.getFusedLocationProviderClient( activity: this);  
  
    mapFragment = (SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map);  
    mapFragment.getMapAsync( onMapReadyCallback: this);  
  
    destinationLatLng = new LatLng( w: 0.0, v1: 0.0);  
  
    mDriverInfo = (LinearLayout) findViewById(R.id.driverInfo);  
  
    mDriverProfileImage = (ImageView) findViewById(R.id.driverProfileImage);  
  
    mDriverName = (TextView) findViewById(R.id.driverName);  
    mDriverPhone = (TextView) findViewById(R.id.driverPhone);  
  
    mRatingBar = (RatingBar) findViewById(R.id.ratingBar);  
  
    mRadioGroup = (RadioGroup) findViewById(R.id.radioGroup);  
    mRadioGroup.check(R.id.UberX);  
  
    mLogout = (Button) findViewById(R.id.logout);  
    mRequest = (Button) findViewById(R.id.request);  
    mSettings = (Button) findViewById(R.id.settings);  
    mHistory = (Button) findViewById(R.id.history);  
}
```

```

mLogout.setOnClickListener((v) -> {
    FirebaseAuth.getInstance().signOut();
    Intent intent = new Intent( packageContext: CustomerMapActivity.this, MainActivity.class);
    startActivity(intent);
    finish();
    return;
});

mRequest.setOnClickListener((v) -> {

    if (requestBol){
        endRide();
    }else{
        int selectId = mRadioGroup.getCheckedRadioButtonId();

        final RadioButton radioButton = (RadioButton) findViewById(selectId);

        if (radioButton.getText() == null){
            return;
        }

        requestService = radioButton.getText().toString();

        requestBol = true;

        String userId = FirebaseAuth.getInstance().getCurrentUser().getUid();

        DatabaseReference ref = FirebaseDatabase.getInstance().getReference( s: "customerRequest");
        GeoFire geoFire = new GeoFire(ref);
        geoFire.setLocation(userId, new GeoLocation(mLastLocation.getLatitude(), mLastLocation.getLongitude()));

        pickupLocation = new LatLng(mLastLocation.getLatitude(), mLastLocation.getLongitude());
        pickupMarker = mMap.addMarker(new MarkerOptions().position(pickupLocation).title("Service Here")
            .icon(BitmapDescriptorFactory.fromResource(R.mipmap.ic_pickup)));

        mRequest.setText("Getting your Mechanic please wait...");

        getClosestDriver();
    }
});

mSettings.setOnClickListener((v) -> {
    Intent intent = new Intent( packageContext: CustomerMapActivity.this, CustomerSettingsActivity.class);
    startActivity(intent);
    return;
});

mHistory.setOnClickListener((v) -> {
    Intent intent = new Intent( packageContext: CustomerMapActivity.this, HistoryActivity.class);
    intent.putExtra( name: "customerOrDriver", value: "Customers");
    startActivity(intent);
    return;
});

PlaceAutocompleteFragment autocompleteFragment = (PlaceAutocompleteFragment)
    fragmentManager().findFragmentById(R.id.place_autocomplete_fragment);

autocompleteFragment.setOnPlaceSelectedListener(new PlaceSelectionListener() {
    @Override
    public void onPlaceSelected(Place place) {
        // TODO: Get info about the selected place.
        destination = place.getName().toString();
        destinationLatLng = place.getLatLng();
    }
});

```



```

    }
    @Override
    public void onError(Status status) {
        // TODO: Handle the error.
    }
});

}

private int radius = 1;
private Boolean driverFound = false;
private String driverFoundID;

GeoQuery geoQuery;
private void getClosestDriver(){...}

/*----- Map specific functions -----
| Function(s) getDriverLocation
|
| Purpose: Get's most updated driver location and it's always checking for movements.
|
| Note:
|     Even tho we used geofire to push the location of the driver we can use a normal
|     Listener to get it's location with no problem.
|
|     0 -> Latitude
|     1 -> Longitude
|
|-----*/

private Marker mDriverMarker;
private DatabaseReference driverLocationRef;
private ValueEventListener driverLocationRefListener;
private void getDriverLocation(){...}

```

```

private void getDriverInfo() {
    mDriverInfo.setVisibility(View.VISIBLE);
    DatabaseReference mCustomerDatabase = FirebaseDatabase.getInstance().getReference().child("Users")
        .child("Drivers").child(driverFoundID);
    mCustomerDatabase.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
            if(dataSnapshot.exists() && dataSnapshot.getChildrenCount()>0){
                if(dataSnapshot.child("name")!=null){
                    mDriverName.setText(dataSnapshot.child("name").getValue().toString());
                }
                if(dataSnapshot.child("phone")!=null){
                    mDriverPhone.setText(dataSnapshot.child("phone").getValue().toString());
                }
                if(dataSnapshot.child("car")!=null){
                    mDriverCar.setText(dataSnapshot.child("car").getValue().toString());
                }
                if(dataSnapshot.child("profileImageUrl").getValue()!=null){
                    Glide.with(getApplication()).load(dataSnapshot.child("profileImageUrl")
                        .getValue().toString()).into(mDriverProfileImage);
                }
                int ratingSum = 0;
                float ratingsTotal = 0;
                float ratingsAvg = 0;
                for (DataSnapshot child : dataSnapshot.child("rating").getChildren()){
                    ratingSum = ratingSum + Integer.valueOf(child.getValue().toString());
                    ratingsTotal++;
                }
                if(ratingsTotal!= 0){
                    ratingsAvg = ratingSum/ratingsTotal;
                    mRatingBar.setRating(ratingsAvg);
                }
            }
        }
    });
}

```

```

private void getDriversAround() {
    getDriversAroundStarted = true;
    DatabaseReference driverLocation = FirebaseDatabase.getInstance().getReference().child("driversAvailable");

    GeoFire geoFire = new GeoFire(driverLocation);
    GeoQuery geoQuery = geoFire.queryAtLocation(new GeoLocation(mLastLocation.getLongitude()
        , mLastLocation.getLatitude()), radius: 999999999);

    geoQuery.addGeoQueryEventListener(new GeoQueryEventListener() {
        @Override
        public void onKeyEntered(String key, GeoLocation location) {

            for(Marker markerIt : markers){
                if(markerIt.getTag().equals(key))
                    return;
            }

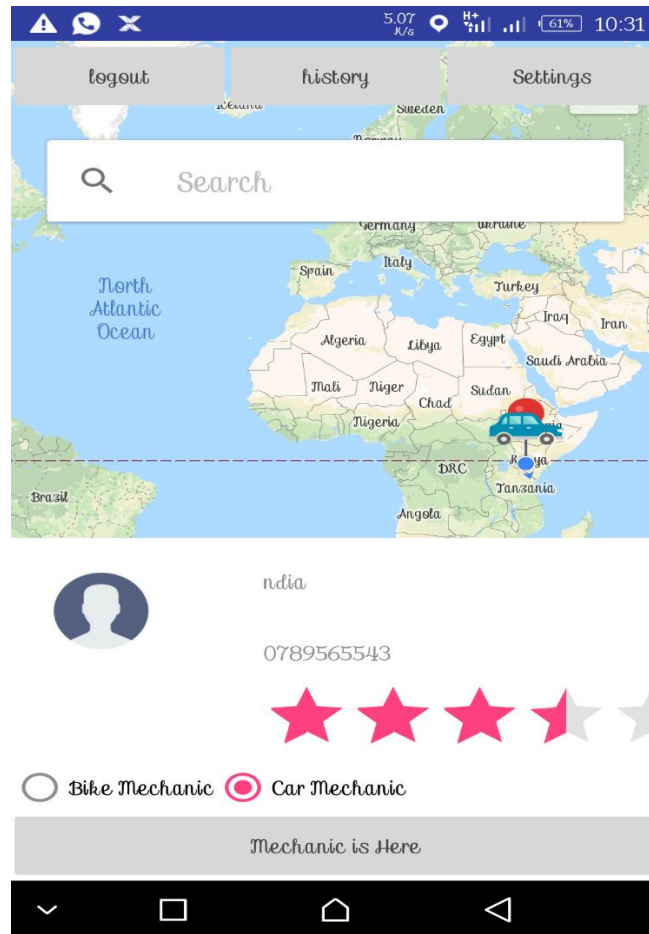
            LatLng driverLocation = new LatLng(location.latitude, location.longitude);

            Marker mDriverMarker = mMap.addMarker(new MarkerOptions().position(driverLocation).title(key)
                .icon(BitmapDescriptorFactory.fromResource(R.mipmap.ic_car)));
            mDriverMarker.setTag(key);

            markers.add(mDriverMarker);
        }
    });
}

```

When the drivers registers for the app it will automatically shows them where they are at that time , the driver is required to enter their names and choose the services they require, for example they can choose whether they need a motorbikes services or the car services then he is require to confirm so that the mechanic will be made available to them depending on the services they have requested. Here is the what it looks like when the driver calls a mechanic.



*Figure 6.6 locating a mechanic*

When the mechanic register for the app he is required to enter the name, phone number and the services he is capable to offer such that when the customer request their services the right customer will send them the right request, they can be able to solve. Here is interface produced when the mechanic signs in, he will receive the notification from the driver requesting him for a service.

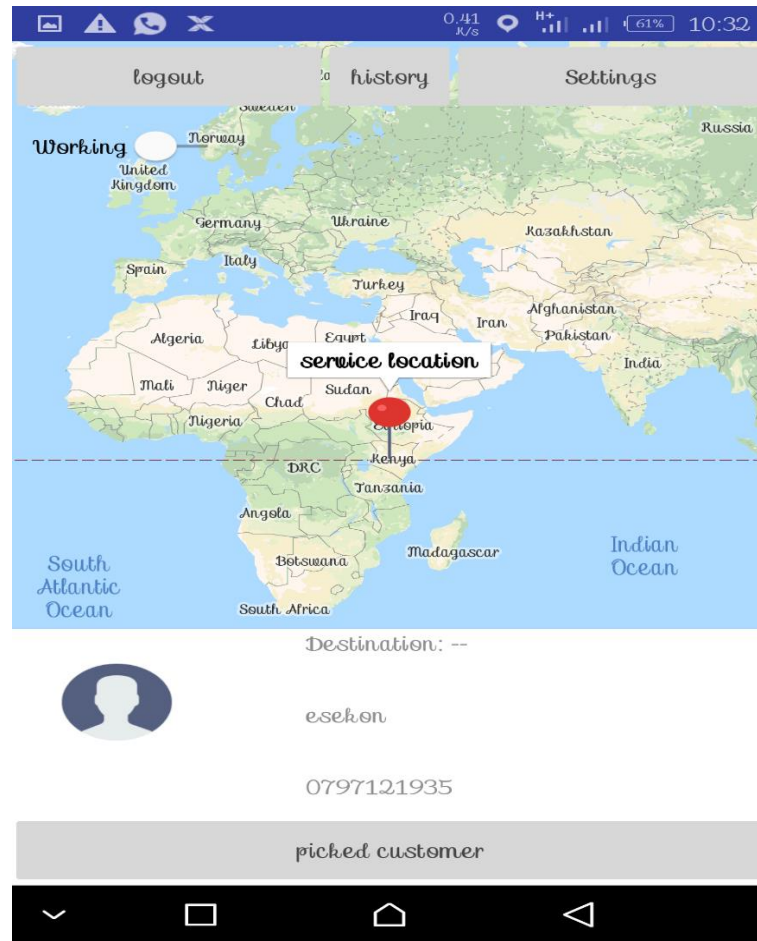


Figure 6.7 Service Request

When the mechanic accepts the service request. This is what will be showed on his maps.

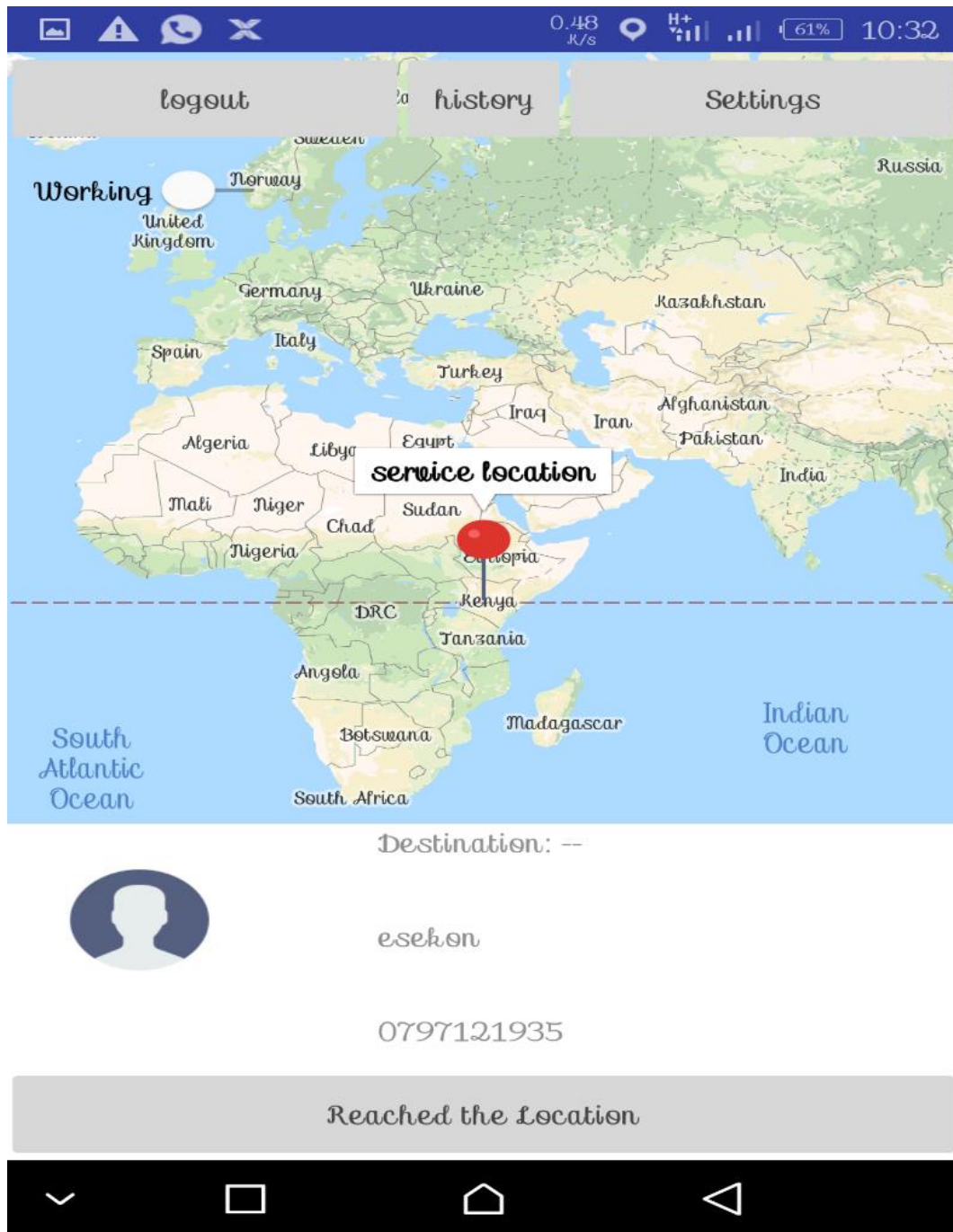


Figure 6.8 accepting service request

## 6.2 TESTING

Validation and input testing

### Test case

Email address:

Password:

Name:

Phone number:

Application testing is a collection of tests designed to verify that a program or the application programs is ready for production. The following is the application testing for the mechanic finder app.

New user registration

When registering a new member, the member must input the correct email and a password that is not less than 6 characters, it may or may not combine multiple characters and must make sure that he/she has inputted the phone number, name and the kind of services he requires failure for this the application could be able to operate as it expected. For instance, this is what will happen when the user did not input the email or the password.



```

mRegistration.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        final String email = mEmail.getText().toString();
        final String password = mPassword.getText().toString();
        if (TextUtils.isEmpty(email)){
            mEmail.setError("please input email to proceed");
            mEmail.requestFocus();

        }else if (TextUtils.isEmpty(password)){
            mPassword.setError("please input your password");
            mPassword.requestFocus();
        }

        else{
            mAuth.createUserWithEmailAndPassword(email, password).addOnCompleteListener( activity: CustomerLoginActivity
                .this, (task) -> {
                    if(!task.isSuccessful()){
                        Toast.makeText( context: CustomerLoginActivity.this, text: "sign up error", Toast.LENGTH_SHORT).show();
                    }else{
                        String user_id = mAuth.getCurrentUser().getUid();
                        DatabaseReference current_user_db = FirebaseDatabase.getInstance()
                            .getReference().child("Users").child("Customers").child(user_id);
                        current_user_db.setValue(true);
                    }
                });
        }
    }
});

```

The above codes to validate and authenticate new member.

```

mLogin.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        final String email = mEmail.getText().toString();
        final String password = mPassword.getText().toString();
        if (TextUtils.isEmpty(email)){
            mEmail.setError("please input your email");
            mEmail.requestFocus();
        }
        else if (TextUtils.isEmpty(password)){
            mPassword.setError("please input your password");
            mPassword.requestFocus();
        }
        else{
            mAuth.signInWithEmailAndPassword(email, password).addOnCompleteListener( activity: CustomerLoginActivity.
                this, (task) -> {
                    if(!task.isSuccessful()){
                        Toast.makeText( context: CustomerLoginActivity.this, text: "sign in error", Toast.LENGTH_SHORT).show();
                    }
                });
        }
    }
});

```

The above codes are used to validate and authenticate the currently user to login to the system.



## **CHAPTER SEVEN: CONCLUSION AND RECOMMENDATION**

### **7.1 CONCLUSION**

Mechanic finder application is at verge race to make the riding exercise along the Kenyan highways to be very easy whenever car breakdown occurs. The long-term implication of this application is already defined because the driver is able to locate the mechanic at time of car breakdown. The driver has to call the mechanic for the services and the mechanic will actually confirmed that by accepting the service request. The main aim of this application was to locate the nearest mechanic to help the driver and this application have really tried to make things better. The application will actually locate the mechanic upon call mechanic function is called and whoever requesting the service is shown to the mechanic.

### **7.2 RECOMMENDATION**

Newly developed application only performs with availability of internet connection or access of internet and android smart phones. Therefore, in the near future we will look whether we can integrate these services to be accessed in USSD format.

The system does not return the driver's destination on the notification screen of the mechanic which has left the mechanic to use pointer location to locate the driver in the map. We recommend to check on this in the near future.

Mechanic finder do not have admin panel for now. The team is working tirelessly to make sure in the near future the mechanic finder application will have admin panel to enable and the admin oversees the operations in the system.

## REFERENCES

1. Chou, (2011). U.S. Patent No. 6,330,499. Washington, DC: U.S. Patent and Trademark Office.
2. Chakra, A. (2017). U.S. Patent No. 6,330,499. Washington, DC: U.S. Patent and Trademark Office.
3. Ombudsman (2016). UK. The Ombudsman Motor B. W., Patent and Trademark Office.
4. White, M. K. (2010). Re-authoring lives: Interviews & essays (pp. 199-213). Adelaide: Dulwich Centre Publications.
5. Guest, G., Bunce, A., & Johnson, L. (2013). How many interviews are enough? An experiment with data saturation and variability. *Field methods*, 18(1), 59-82.
6. Boehm, B. (2011). A spiral model of software development and enhancement. *ACM SIGSOFT Software engineering notes*, 11(4), 14-24.
7. Govokhina, O., Baily, G., & Breton, G. (2014, August). Learning optimal audio-visual phasing for a HMM-based control model for facial animation. In 6th ISCA Workshop on Speech Synthesis (SSW6) (pp. 1-4).
8. Sudman, S., Bradburn, N. M., & Schwarz, N. (2016). Thinking about answers: The application of cognitive processes to survey methodology.
9. Giese, H., & Wagner, R. (2006, October). Incremental model synchronization with triple graph grammars. In *International Conference on Model Driven Engineering Languages and Systems* (pp. 543-557). Springer, Berlin, Heidelberg.
10. Fenzi. (1971). Washington, DC:US. Patent and Trademark Office.
11. Zykov, S. V., Gromoff, A., & Kazantsev, N. S. (Eds.). (2018). *Software Engineering for Enterprise System Agility: Emerging Research and Opportunities: Emerging Research and Opportunities*. IGI Global.
12. Barber L B (2008). *Procedure of data collection. journal*, 45(8)
13. Cochran, W. G. (1977). *Sampling Techniques: 3d Ed*. New York: Wiley.
14. Sangiovanni-Vincentelli, A., & Martin, G. (2001). Platform-based design and software design methodology for embedded systems. *IEEE Design & Test of Computers*, 18(6), 23-3
15. Tarhini, A., Yunis, M., & El-Kassar, A. N. (2018). Innovative sustainable methodology for managing in-house software development in SMEs. *Benchmarking: An International Journal*, 25(3), 1085-1103.

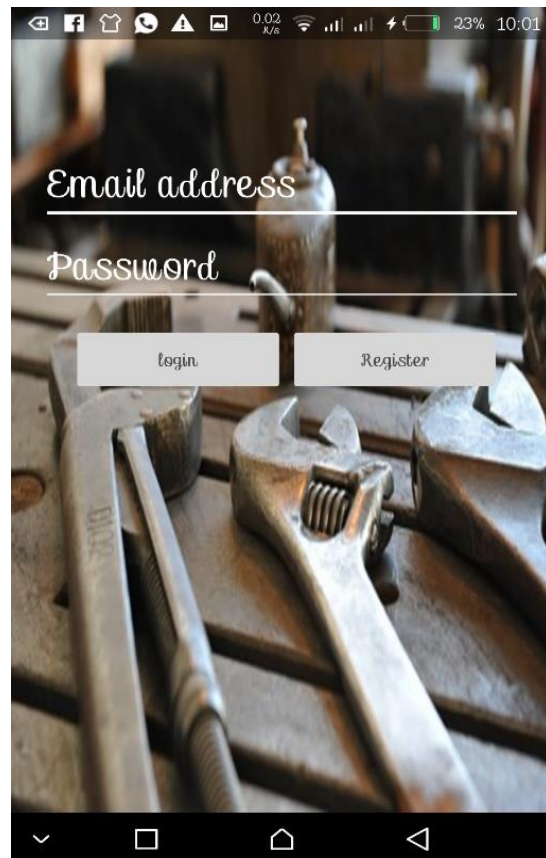
## APPENDICES

### APPENDIX I: USER MANUAL

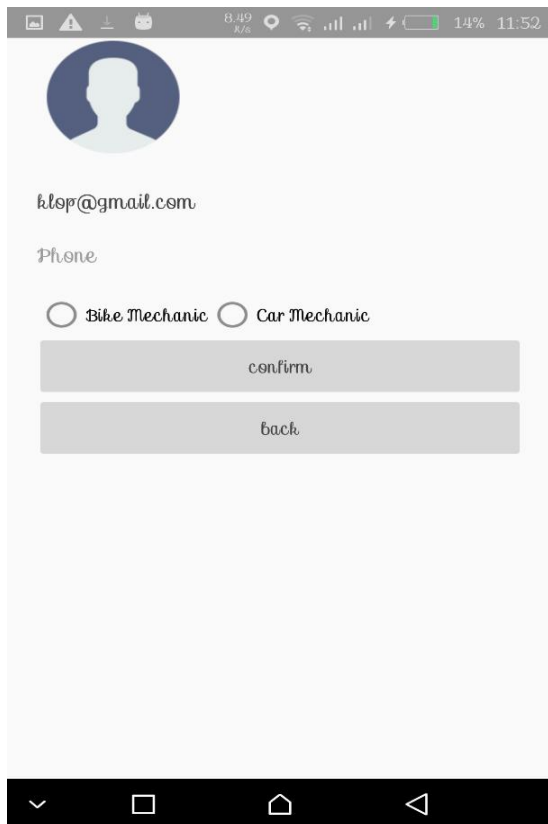
In the home page the app user is given two options to make, either to be a mechanic or a customer(driver). Upon making the decision you will be redirected where you can sign in if you are already a user or register if you are a new member. The user is required to confirm whatever the services he is capable of delivering in case he/she has chosen to be a mechanic and for the customer he has to choose whatever the services he required to get and then he will go ahead to make a call to the mechanic.



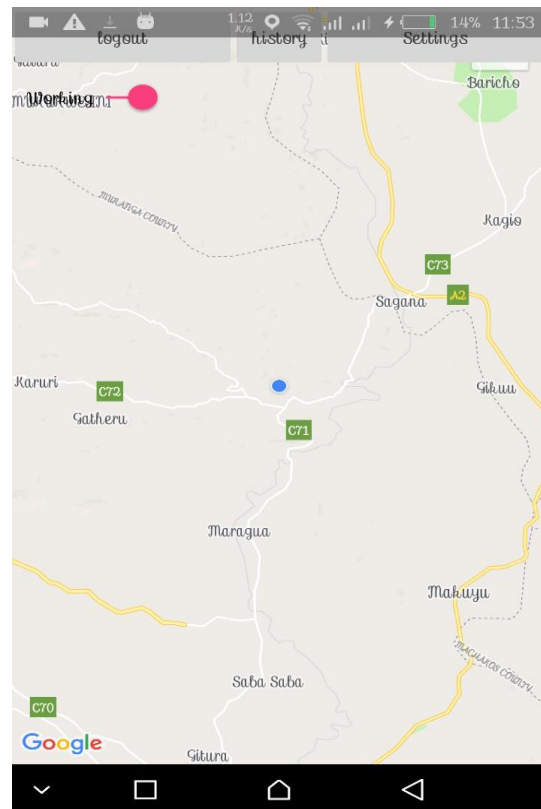
*App interface*



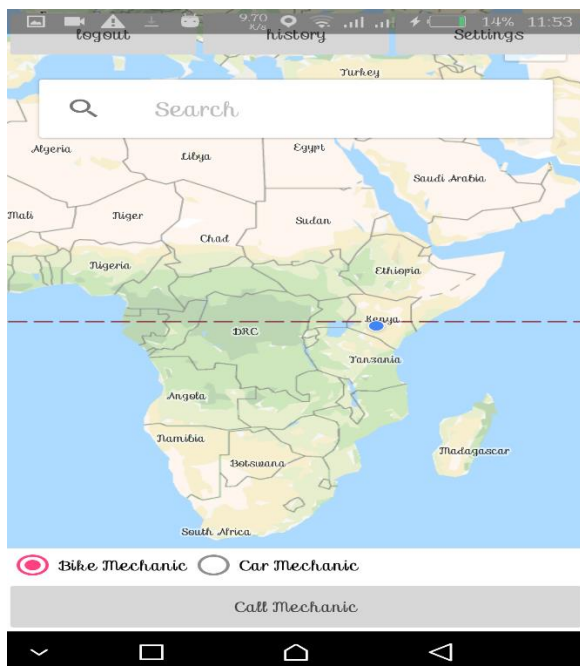
*user registration and login interface*



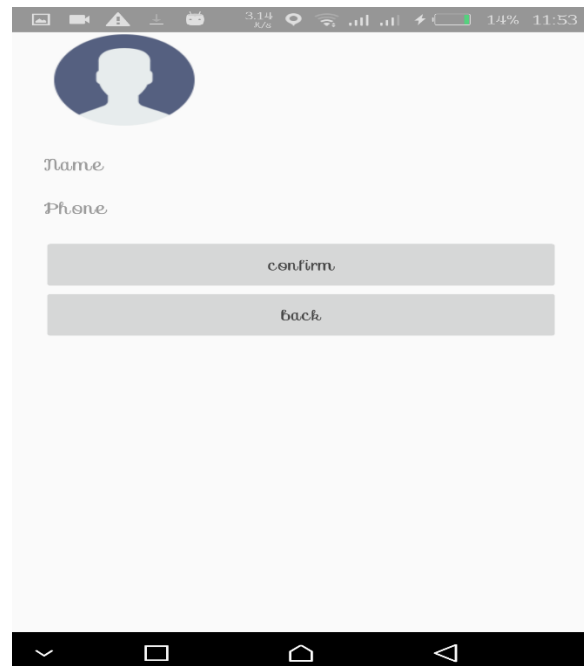
*Mechanic confirmation interface*



*mechanic location interface*



*Driver location interface*



*driver confirmation interface.*


## APPENDIX 2: INSTALLATION GUIDE.

During app installation in android studio, set up your devices as follows;

1. Connect your device to your development machine with a USB cable. If you are developing on windows, you might need to install the appropriate USB driver for your device.
2. Enable USB debugging in the Developer options as follows
  - ✓ Open the setting app
  - ✓ (only on android 8.0 or higher) select system
  - ✓ Scroll to the bottom and select about phone
  - ✓ Scroll to the bottom and tap builder number 7 times
  - ✓ Return to the previous screen to find developer options near the bottom.

Open developer options, and the scroll down to find and enable USB debugging.

### Run the app on your device as follows.

1. In android studio, click the app module in the project window and then select Run  in the toolbar.
2. In the select deployment Target window, select your device and click ok.

Android studio installs the app on your connected device and starts it. you should now see the app running on your device after a successfully build up

### Run on an emulator;

1. In android studio, click the app module n the project window and the select run >run in the toolbar.
2. In the select deployment target window, click create new virtual device.
3. In the select hardware screen, select a phone device, such as pixel and the click next.
4. In the system image, select the version with the highest API level. If you don't have the version installed, a download link is shown, so click that and complete the download
5. Click next
6. On the android virtual device (AVD) screen, leave all the settings alone and click finish.
7. Back in the select deployment target dialog, select the device you just created and click **OK**.

Android studio installs the app on the emulator and starts it. You should now see the app running on the emulator.

### **APPENDIX 3: QUESTIONNAIRE**

**Directions: Please try to answer this question genuinely.**

**I Please fill in your details here,** (notes; your data will not be disclosed to anyone else. Your privacy is our esteem concern)

Name:

Age:

Gender:

Email Address:

#### **Questions**

Q1: what is your experience riding on Kenyan roads?

Q2: Have you ever got stuck on the road due to car breakdown for long hours without being attended to?

☐ Yes ☐ No

If yes? Please gives us a reason why?

Q3: How would you like this problem to be solved?

Q4: Do you think coming up with a mobile application to locate a mechanic will do better in solving this problem?

☐ Yes ☐ No. if yes, why?

**Thank you for sharing your thoughts with us.**