**1. INTRODUCTION**

The Vehicle Location Tracker And Speed Notifier Using Android Mobile Application is designed to track the vehicle and notify the speed. One of the main innovative features of this project includes the usage of the phenomenon called INTERNET OF THINGS. This phenomenon allows the application users to track the vehicle location at anywhere and anytime.

## 1.1 Document Purpose:

The current document describes about the Software Requirement Specifications for the Project “DESIGN AND IMPLEMENTATION OF THE VEHICLE LOCATION TRACKER AND SPEED NOTIFIER” which is an android mobile application. This application is used to track the vehicle location and notify the speed limits at "anytime and anywhere".

## 1.2 Project Scope:

We find many situations in day to day life where it is necessary to find the vehicle location in an emergency situations and also want to know at what speed the vehicle is travelling,With day-by-day advancing technology, we are given easy access to this situation.Yes! Here it is. Our project targets to develop an Android mobile application which provides the user the facility to access the vehicle location and speed at which he/she is travelling. The project works over internet which allows the users to know the vehicle location.

###### **1.3 Problem Statement :**

The "Vehicle Location Tracker and Speed Notifier is the android mobile application", which intimates the guardian about the vehicle location and also the speed at which user is travelling by using GPS system in an application.

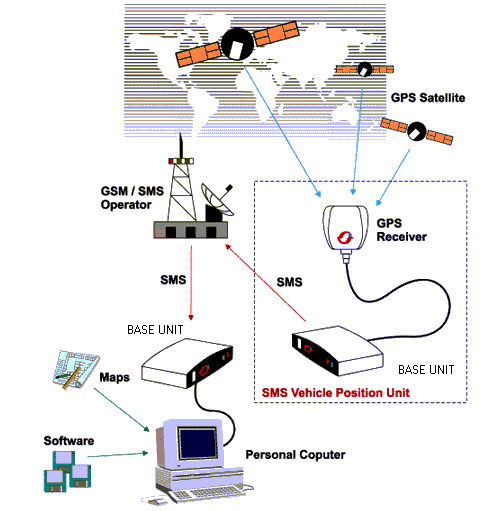
## 1.4 Document Conventions:

Bond paper should be used for the preparation of final document. Typing should be done on the 12 point size letters for the running text, 14 point size for the sub-headings and 16 point size for main heading/titles/names/etc. The font should be preferably Times New Roman.

**2. LITERATURE SURVEY**

**2.1. Existing System :**

* The basic system which still exists should have an gps receiver, base unit, personal computer and all stuff which is used to track the vehicle location using gps system as shown in figure2.1.it is not automatic tracking system.



**Fig:2.1.1 existing system**

**2.2. Proposed System:**

To eliminate the cases that one encounters in the existing system, we are making use of the advancing technology, introducing the mobile application for tracking the location which doesn’t require any of the components. It directly track the location from the mobile automatically and also notifies the speed.

## 2.3.Project Perspective:

The person uses the android mobile application to track the vehicle location. This mobile application is very useful when people are in emergency and need help, situations like accidents, in such cases by sending the location of that vehicle to the parent user. It mainly works on GPS and Internet systems. Vehicle is traced by using information from satellite systems and send to GSM systems.  Vehicle and user are connected initially by using GSM network. User will send message to Moving vehicle for connectivity, hardware mounted on vehicle reacts to message and send conformation message to user and then if the message is valid one then GPS modem is initiated. The hardware interface responds to the signal and performs the specified action, i.e tracking the vehicle location and also notify the speed to the guardian user.

This system proposes the objective, to implement it in areas where access should become “easy & handy”, provide “security” and mainly, provide access from “anywhere-anytime”. This project uses internet as its key element and the concept used is called as “INTERNET OF THINGS”.

**Internet of things –**

* It relates to performing physical work automatically.
* It can be described as a point where the web and the physical world meet.
* It involves connecting everyday objects like smart-phones, Internet TVs, sensors and actuators to the Internet where the devices are intelligently linked together enabling new forms of communication between things and people, and between things themselves.

**3. REQUIRIMENT ANALYSIS**

**3.1 Hardware Specification:**

* Processor : Pentium-IV 2.6 GHz
* Ram : 512MB
* Monitor : 15”
* Hard disk : 20GB
* CD drive : 52X
* Keyboard : Standard 102 keys

**3.2 Software Specification:**

* Eclipse IDE
* SQLite
* Java
* XML

**3.3 Feasibility Study:**

In this feasibility study phase we have undergone various steps which are described below:

1. Identify the origin of the information at different level.

2. Should identify the expectation of user from computerized system.

3. Analyze the drawback of existing system (manual) system.

Three key considerations involved in the feasibility analysis are

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

**3.3.1 Economical Feasibility:**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

**3.3.2 Technical Feasibility:**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**3.3.3 Social Feasibility:**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

## 3.4 Safety and Security Requirements :

As you are likely aware, every time that you download and install any “normal” application, we are risking virus threat. Prior to java, most users did not download executable programs frequently, and those who did scan them for viruses prior to execution. Even so, most users still worried about the possibility of infecting their system with a virus. When you use java-compatible applicationsyou can safely download and install Mobile applications, without fear of virus threats or malicious intent. Java achieves this protection by confining a java program to the java execution environment and not allowing it access to other parts of computer.

**4.SYSTEM DESIGN**

**4.1 Unified Modeling Language:**

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

* + **User Model View**
    1. This view represents the system from the users perspective.
    2. The analysis representation describes a usage scenario from the end-users perspective.
  + **Structural model view**
    1. In this model the data and functionality are arrived from inside the system.
    2. This model view models the static structures.
* **Behavioral Model View**

It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

* **Implementation Model View**

In this the structural and behavioral as parts of the system are represented as they are to be built.

* **Environmental Model View**

In this the structural and behavioral aspects of the environment in which the system is to be implemented are represented

**UML is specifically constructed through two different domains they are:**

UML Analysis modeling, this focuses on the user model and structural model views of the system.UML design modeling, which focuses on the behavioral modeling, implementation modeling and environmental model views

Use case Diagrams represent the functionality of the system from a user’s point of view. Use cases are used during requirements elicitation and analysis to represent the functionality of the system. Use cases focus on the behavior of the system from external point of view. Actors are external entities that interact with the system. Examples of actors include users like administrator, bank customer …etc., or another system like central database.

**5. UML DIAGRAMS**

**5.1 Class Diagram:**

Class diagram show cases the classes involved, their attributes and their functionalities and the relationships among objects.



**Fig:5.1.1class diagram**

**Description:**

This class diagram consists of classes –.

|  |  |  |
| --- | --- | --- |
|  | **ATTRIBUTES** | **FUNCTIONALITIES** |
| **User** | 1. Name 2. id 3. user location | 1.sending msg to parent()  2.fetching phone no from database()  3.getting location details() |
| **Parent** | 1.Name  2.mobile no | 1.accessing child location and speed() |
| **Mobile** | 1.mobile type  2.id  3.mobile number | 1.sending and fetching data() |
| **Sqlitedb** | 1.database type  2.database id | 1.providingdata()  2.fetching and sending data() |

**5.2 Usecase Diagram:**

Use case diagram consists of actors and use cases and represents the relationship between them.



**Fig:5.2.1 usecase diagram**

**Description for usecase:**

Internet connection establishment: Firstly the user need to connect to the internet inorder to access the android mobile app

User : Inorder to access the android mobile app, the user install to the app.

Send message: request for the vehicle location is sent to the GPS.

Database : the request is accepted by the database and the specified location is sent with longitude and latitude values.

Parent message: the tracked location can automatically send to the parent by normal text message.

**5.3 Sequence Diagram:**

Sequence Diagram represents the objects participating in the interaction horizontally and time vertically.

This sequence diagram consists of –

****

**Fig.5.3.1 sequence diagram**

**Description for sequence diagram:**

Firstly the user registers with the app by creating a password, and then logins into the app. Then the android mobile app authenticates the user through the server.

The location is tracked automatically by clicking on the button and then the longitude latitude values are displayed .

The longitude latitude values given can be see in maps for the specified location

Ths tracking location and speed of vehicle are sent to the parent user.

**5.4 Collaboration Diagram:**

Collaboration diagram describes the organization of messages between the objects.

This diagram consists of objects -



**Fig 5.4.1 collabration diagram**

**Description for collaboration diagram**:

Firstly the user registers with the app by creating a password, and then logins into the app. Then the android mobile app authenticates the user through the server.

Then the android mobile app authenticates the user through the server.

The location is tracked automatically by clicking on the button and then the longitude latitude values are displayed .

The longitude latitude values given can be see in maps for the specified location

Ths tracking location and speed of vehicle are sent to the parent user.

**5.5 Activity Diagram:**

Activity diagrams provide a way to model the workflow of a business process. You can also use activity diagrams to model code-specific information such as a class operation. Activity diagrams are very similar to a flowchart because you can model a workflow from activity to activity.

Activity Diagram Tools*:*You can use the following tools on the activity diagram toolbox to model activity diagrams:

* Start states
* States
* Swim lanes
* Synchronizations
* Transmissions
* Activities
* Decisions
* End state
* Object
* Object Flow

This diagram show cases the flow of the project works.



**Fig 5.5.1 activity diagram**

**Description for Activity diagram**:

The application connects to the internet.

User connects to the app by registering to the app if its successful then the user can track the location and also sends the location to the specified user.

If it fails the error msg will be indicated

This application can also tell the user where he/she is exactly present.

**5.6 Statechart Diagram:**

It describes different states of a component in a system. The states are specific to a component/object of a system.



**Fig.5.6.1 statechart diagram**

**Description for statechart diagram:**

The application connects to the internet.

User connects to the app by registering to the app if its successful then the user can track the location and also sends the location to the specified user.

If it fails the error msg will be indicated

This application can also tell the user where he/she is exactly present.

**5.7 Component Diagram:**

Component diagrams are used to model physical aspects of a system. Physical aspects are the elements like executable, libraries, files, documents etc., which resides in a node. So component diagrams are used to visualize the organization and relationships among components in a system. These diagrams are also used to make executable systems.

****

**Fig:5.7.1 component diagram**

The above component diagram shows different components like user, android mobile app, data base ,parent user and also the association between them.

**5.8 Deployment Diagram:**

Deployment diagram are used to visualize the topology of the physical components of a system where the software components are deployed. So deployment diagram are used to describe the static deployment view of a system. Deployment diagram consists of nodes and their relationships.

****

**Fig:5.8.1 deployment diagram**

The above deployment diagram consists of nodes – user, android mobile app, databse,parent mobile. The information about the nodes can be explained through notes. For example in the above diagram ,the functionality of the user is to access the app. And the same is depicted through note. This deployment diagram describes the static deployment view of a system.

**6. IMPLEMENTATION**

**6.1 Problem Definition:**

Inorder to track the vehicle location automatically this mobile application uses the GPS system connectivity .The "Vehicle Location Tracker and Speed Notifier is the android mobile application", which intimates the guardian about the vehicle location and also the speed at which user is travelling by using GPS system in an application.

**6.2 Modules Description:**

Number of Modules after careful analysis the system has been identified to have the following modules:

* **User Authentication Module**:

The user get registered with the application by installing theapp ,while registering he/she should specify the alternate number for an emergency contact.

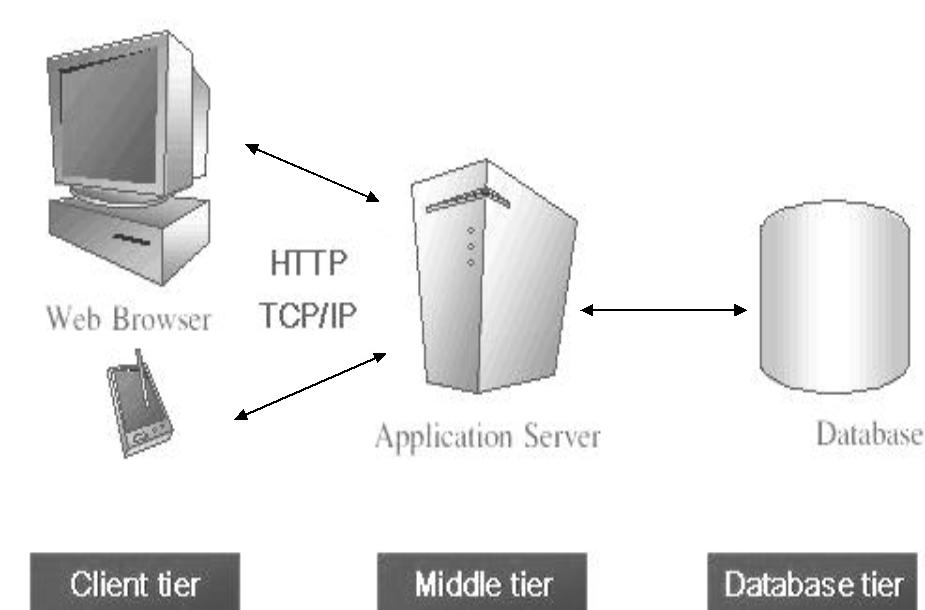
* **Vehicle location tracker:**

After installing the app , the user can know location of the vehicle from any where and at any time by using the internet .

* **Normal message:**

The normal message is sent to the specified user with the tracking location and speed limit details.

**6.3 SYSTEM ARCHITECTURE**



1. **THE CLIENT TIER:**

Also called as the client layer comprises of components that are dedicated to presenting the data to the user. For example: Windows/Web Forms and buttons, edit boxes, Text boxes, labels, grids, etc.

1. **THE MIDDLE LAYER**

This layer encapsulates the Business rules or the business logic of the encapsulations. To have a separate layer for business logic is of a great advantage. This is because any changes in Business Rules can be easily handled in this layer. As long as the interface between the layers remains the same, any changes to the functionality/processing logic in this layer can be made without impacting the others. A lot of client-server apps failed to implement successfully as changing the business logic was a painful process

1. **THE DATA BASE LAYER**

This layer comprises of components that help in accessing the Database. If used in the right way, this layer provides a level of abstraction for the database structures. Simply put changes made to the database, tables, etc do not affect the rest of the application because of the Data Access layer. The different application layers send the data requests to this layer and receive the response from this layer.

1. **THE DATABASE LAYER**

This layer comprises of the Database Components such as DB Files, Tables, Views, etc. The Actual database could be created using SQL Server, Oracle, Flat files, etc. In an n-tier application, the entire application can be implemented in such a way that it is independent of the actual Database. For instance, you could change the Database Location with minimal changes to Data Access Layer. The rest of the Application should remain unaffected.

**6.4 Technologies Used:**

**6.4.1 Android**

**Android Operating System:**

Android is a mobile operating system (OS) based on the Linux kernel and currently developed by Google. With a user interface based on direct manipulation, Android is designed primarily for touch screen mobile devices such as smart phones and tablet computers, with specialized user interfaces for televisions (Android TV), cars (Android Auto), and wrist watches (Android Wear). The OS uses touch inputs that loosely correspond to real-world actions, like swiping, tapping, pinching, and reverse pinching to manipulate on-screen objects, and a virtual keyboard. Despite being primarily designed for touchscreen input, it also has been used in game consoles, digital cameras, regular PCs (e.g. the HP Slate 21) and other electronics.

Android, Inc. was founded in Palo Alto, California in October 2003 by Andy Rubin (co-founder of Danger), Rich Miner (co-founder of Wildfire Communications, Inc.),Nick Sears (once VP at T-Mobile),and Chris White (headed design and interface development at WebTV) to develop, in Rubin's words, "smarter mobile devices that are more aware of its owner's location and preferences". The early intentions of the company were to develop an advanced operating system for digital cameras. Though, when it was realized that the market for the devices was not large enough, the company diverted its efforts toward producing a smart phone operating system that would rival Symbian and Microsoft Windows Mobile.

Google acquired Android Inc. on August 17, 2005; key employees of Android Inc., including Rubin, Miner, and White, stayed at the company after the acquisition.Not much was known about Android Inc. at the time, but many assumed that Google was planning to enter the mobile phone market with this move. At Google, the team led by Rubin developed a mobile device platform powered by the Linux kernel. Google marketed the platform to handset makers and carriers on the promise of providing a flexible, upgradable system. Google had lined up a series of hardware component and software partners and signaled to carriers that it was open to various degrees of cooperation on their part.

**Android Applications:**

Applications ("apps"), that extend the functionality of devices, are written primarily in the Java programming language (without the usual "write once, run anywhere" claim of the Java platform) using the Android software development kit (SDK). The SDK includes a comprehensive set of development tools, including a debugger, software libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials. Initially, Google's supported integrated development environment (IDE) was Eclipse using the Android Development Tools (ADT) plugin ; in December 2014, Google released Android Studio, based on IntelliJ IDEA, as its primary IDE for Android application development. Other development tools are available, including a Native Development Kit for applications or extensions in C or C++, Google App Inventor, a visual environment for novice programmers, and various cross platform mobile web applications frameworks. In January 2014, Google unveiled an Apache Cordova–based framework for porting Chrome HTML 5 applications to Android, wrapped in a native application shell.

Android has a growing selection of third-party applications, which can be acquired by users by downloading and installing the application's APK file, or by downloading them using an application store program that allows users to install, update, and remove applications from their devices. Google Play Store is the primary application store installed on Android devices that comply with Google's compatibility requirements and license the Google Mobile Services software. Google Play Store allows users to browse, download and update applications published by Google and third-party developers; As of July 2013, there are more than one million applications available for Android in Play Store. As of May 2013, 48 billion applications have been installed from Google Play Store and in July 2013, 50 billion applications were installed. Some carriers offer direct carrier billing for Google Play application purchases, where the cost of the application is added to the user's monthly bill.

Due to the open nature of Android, a number of third-party application marketplaces also exist for Android, either to provide a substitute for devices that are not allowed to ship with Google Play Store, provide applications that cannot be offered on Google Play Store due to policy violations, or for other reasons. Examples of these third-party stores have included the Amazon Appstore, GetJar, and SlideMe. F-Droid, another alternative marketplace, seeks to only provide applications that are distributed under free and open source licenses.

**6.4.2 Java**

Java is an object oriented programming language. Writing object-oriented programs involves creating classes, creating objects from those classes, and creating applications, which are stand-alone executable programs that use those objects.

A class is a template, blueprint, or contract that defines what an object’s data fields and methods will be. An object is an instance of a class. You can create many instances of a class. A Java class uses variables to define data fields and methods to define actions. Additionally, a class provides methods of a special type, known as constructors, which are invoked to create a new object. A constructor can perform any action, but constructors are designed to perform initializing actions, such as initializing the data fields of objects.

Objects are made up of attributes and methods. Attributes are the characteristics that define an object; the values contained in attributes differentiate objects of the same class from one another.

There are three main features of OOPS.

**1)** Encapsulation

**2)** Inheritance

**3)** Polymorphism

**Encapsulation:**

Encapsulation means putting together all the variables (instance variables) and the methods into a single unit called Class. It also means hiding data and methods within an Object. Encapsulation provides the security that keeps data and methods safe from inadvertent changes. Programmers sometimes refer to encapsulation as using a “black box,” or a device that you can use without regard to the internal mechanisms. A programmer can access and use the methods and data contained in the black box but cannot change them.

**Inheritance:**

It is the ability to create classes that share the attributes and methods of existing classes, but with more specific features. Inheritance is mainly used for code reusability. So you are making use of already written class and further extending on that. That why we discussed about the code reusability the concept. In general one line definition we can tell that deriving a new class from existing class, it’s called as Inheritance.

**Polymorphism:**

Polymorphism definition is that Poly means many and morphos means forms. It describes the feature of languages that allows the same word or symbol to be interpreted correctly in different situations based on the context. There are two types of Polymorphism available in Java. For example, in English the verb “run” means different things if you use it with “a footrace,” a “business,” or “a computer.” You understand the meaning of “run” based on the other words used with it. Object-oriented programs are written so that the methods having same name works differently in different context. Java provides two ways to implement polymorphism.

1. Static Polymorphism (Compile time polymorphism/ Method overloading)
2. Dynamic Polymorphism ( Run time polymorphism / Method overriding)

***Static Polymorphism:***

The ability to execute different method implementations by altering the argument used with the method name is known as method overloading. In below program we have three print methods each with different arguments. When you properly overload a method, you can call it providing different argument lists, and the appropriate version of the method executes

***Dynamic Polymorphism:***

When you create a subclass by extending an existing class, the new subclass contains data and methods that were defined in the original super class. In other words, any child class object has all the attributes of its parent. Sometimes, however, the super class data fields and methods are not entirely appropriate for the subclass objects; in these cases, you want to override the parent class members.

**Abstraction:**

An essential element of object-oriented programming is abstraction. Humans manage complexity through abstraction. When you drive your car you do not have to be concerned with the exact internal working of your car(unless you are a mechanic). What you are concerned with is interacting with your car via its interfaces like steering wheel, brake pedal, accelerator pedal etc. Various manufacturers  of car has different implementation of car working but its basic interface has not changed. Hence the knowledge you have of your car is abstract.

An abstract class is something which is incomplete and you can not create instance of abstract class. If you want to use it you need to make it complete or concrete by extending it. A class is called concrete if it does not contain any abstract method and implements all abstract method inherited from abstract class or interface it has implemented or extended. By the way Java has concept of abstract classes, abstract method but a variable can not be abstract in Java.

**6.4.3 Eclipse:**

In [computer programming](http://en.wikipedia.org/wiki/Computer_programming), **Eclipse** is an [integrated development environment](http://en.wikipedia.org/wiki/Integrated_development_environment) (IDE). It contains a base [workspace](http://en.wikipedia.org/wiki/Workspace) and an extensible [plug-in](http://en.wikipedia.org/wiki/Plug-in_(computing)) system for customizing the environment. Written mostly in [Java](http://en.wikipedia.org/wiki/Java_(programming_language)), Eclipse can be used to develop applications. By means of various plug-ins, Eclipse may also be used to develop applications in other [programming languages](http://en.wikipedia.org/wiki/Programming_language): [Ada](http://en.wikipedia.org/wiki/Ada_(programming_language)), [ABAP](http://en.wikipedia.org/wiki/ABAP), [C](http://en.wikipedia.org/wiki/C_(programming_language)), [C++](http://en.wikipedia.org/wiki/C%2B%2B), [COBOL](http://en.wikipedia.org/wiki/COBOL), [Fortran](http://en.wikipedia.org/wiki/Fortran), [Haskell](http://en.wikipedia.org/wiki/Haskell_(programming_language)), [JavaScript](http://en.wikipedia.org/wiki/JavaScript), [Lasso](http://en.wikipedia.org/wiki/Lasso_(programming_language)), [Lua](http://en.wikipedia.org/wiki/Lua_(programming_language)), [Natural](http://en.wikipedia.org/wiki/NATURAL), [Perl](http://en.wikipedia.org/wiki/Perl), [PHP](http://en.wikipedia.org/wiki/PHP),[Prolog](http://en.wikipedia.org/wiki/Prolog), [Python](http://en.wikipedia.org/wiki/Python_(programming_language)), [R](http://en.wikipedia.org/wiki/R_(programming_language)), [Ruby](http://en.wikipedia.org/wiki/Ruby_(programming_language)) (including [Ruby on Rails](http://en.wikipedia.org/wiki/Ruby_on_Rails) framework), [Scala](http://en.wikipedia.org/wiki/Scala_(programming_language)), [Clojure](http://en.wikipedia.org/wiki/Clojure), [Groovy](http://en.wikipedia.org/wiki/Groovy_(programming_language)), [Scheme](http://en.wikipedia.org/wiki/Scheme_(programming_language)), and [Erlang](http://en.wikipedia.org/wiki/Erlang_(programming_language)). It can also be used to develop packages for the software [Mathematica](http://en.wikipedia.org/wiki/Mathematica). Development environments include the Eclipse Java development tools (JDT) for Java and Scala, Eclipse CDT for C/C++ and Eclipse PDT for PHP, among others.

The initial [codebase](http://en.wikipedia.org/wiki/Codebase) originated from [IBM VisualAge](http://en.wikipedia.org/wiki/IBM_VisualAge). The Eclipse [software development kit](http://en.wikipedia.org/wiki/Software_development_kit) (SDK), which includes the Java development tools, is meant for Java developers. Users can extend its abilities by installing plug-ins written for the Eclipse Platform, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules.

**History:**

Eclipse began as a [Smart Canada](http://en.wikipedia.org/w/index.php?title=Smart_Canada&action=edit&redlink=1) project. [Object Technology International](http://en.wikipedia.org/wiki/Object_Technology_International) (OTI), which had previously marketed the [Smalltalk](http://en.wikipedia.org/wiki/Smalltalk)-based [VisualAge](http://en.wikipedia.org/wiki/VisualAge) family of [integrated development environment](http://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) products, developed the new product as a Java-based replacement. In November 2001, a consortium was formed with a board of stewards to further the development of Eclipse as open-source software. It is estimated that IBM had already invested close to $40 million by that time. The original members were [Borland](http://en.wikipedia.org/wiki/Borland), [IBM](http://en.wikipedia.org/wiki/IBM), [Merant](http://en.wikipedia.org/wiki/Micro_Focus_International), [QNX Software Systems](http://en.wikipedia.org/wiki/QNX_Software_Systems), [Rational Software](http://en.wikipedia.org/wiki/Rational_Software), [Red Hat](http://en.wikipedia.org/wiki/Red_Hat), [SuSE](http://en.wikipedia.org/wiki/SuSE), [Together Soft](http://en.wikipedia.org/wiki/Borland_Together) and [WebGain](http://en.wikipedia.org/wiki/WebGain). The number of stewards increased to over 80 by the end of 2003. In January 2004, the [Eclipse Foundation](http://en.wikipedia.org/wiki/Eclipse_Foundation) was created.

Eclipse 3.0 (released on 21 June 2004) selected the [OSGi](http://en.wikipedia.org/wiki/OSGi) Service Platform specifications as the runtime architecture.

The [Association for Computing Machinery](http://en.wikipedia.org/wiki/Association_for_Computing_Machinery) recognized Eclipse with the 2011 [ACM Software Systems Award](http://en.wikipedia.org/wiki/ACM_Software_Systems_Award) on 26 April 2012.

**6.5 Coding:**

**6.5.1 XML:**

It is the front end design code for mobile application.

**activity\_main.xml :**

<RelativeLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

xmlns:tools=*"http://schemas.android.com/tools"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:paddingBottom=*"@dimen/activity\_vertical\_margin"*

android:paddingLeft=*"@dimen/activity\_horizontal\_margin"*

android:paddingRight=*"@dimen/activity\_horizontal\_margin"*

android:paddingTop=*"@dimen/activity\_vertical\_margin"*

tools:context=*"com.surendra.parkingvehicle.MainActivity"*

android:background=*"@drawable/back01"* >

<RelativeLayout

android:id=*"@+id/rel"*

android:layout\_height=*"wrap\_content"*

android:layout\_width=*"wrap\_content"*

android:layout\_centerInParent=*"true"*>

</RelativeLayout>

<Button

android:id=*"@+id/parking"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignParentLeft=*"true"*

android:layout\_alignParentRight=*"true"*

android:layout\_alignParentTop=*"true"*

android:layout\_marginTop=*"60dp"*

android:text=*"Location"*

/>

<Button

android:id=*"@+id/getTrackikng"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignLeft=*"@+id/parking"*

android:layout\_alignRight=*"@+id/parking"*

android:layout\_below=*"@+id/parking"*

android:layout\_marginTop=*"20dp"*

android:text=*"Tracking"*

/>

<Button

android:id=*"@+id/sms"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignLeft=*"@+id/getTrackikng"*

android:layout\_alignRight=*"@+id/getTrackikng"*

android:layout\_alignTop=*"@+id/rel"*

android:text=*"SMS"* />

<Button

android:id=*"@+id/Activate"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_above=*"@+id/rel"*

android:layout\_alignLeft=*"@+id/sms"*

android:layout\_alignRight=*"@+id/sms"*

android:layout\_marginTop=*"100dp"*

android:text=*"Activate"* />

</RelativeLayout>

**activity\_navigation.xml:**

<RelativeLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

xmlns:tools=*"http://schemas.android.com/tools"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

tools:context=*".NavigationActivity"* >

<fragment

android:id=*"@+id/map"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:layout\_below=*"@+id/back"*

class=*"com.google.android.gms.maps.SupportMapFragment"* />

<Button

android:id=*"@+id/back"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_centerHorizontal=*"true"*

android:layout\_alignParentTop=*"true"*

android:layout\_margin=*"5dp"*

android:text=*"Back"*

android:visibility=*"visible"* />

</RelativeLayout>

**register.xml**

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:orientation=*"vertical"*

android:background=*"@drawable/back"* >

<TextView

android:id=*"@+id/textView1"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_marginLeft=*"35dp"*

android:layout\_marginTop=*"40dp"*

android:text=*"Registration"*

android:textSize=*"50dp"* />

<EditText

android:id=*"@+id/name"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:ems=*"10"*

android:layout\_marginTop=*"30dp"*

android:layout\_marginLeft=*"30dp"*

android:layout\_marginRight=*"30dp"*

android:hint=*" Enter Name"*

android:gravity=*"center"*

android:inputType=*"textPersonName"* >

</EditText>

<EditText

android:id=*"@+id/phone"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:ems=*"10"*

android:layout\_marginLeft=*"30dp"*

android:layout\_marginTop=*"30dp"*

android:layout\_marginRight=*"30dp"*

android:hint=*" Enter Phone"*

android:gravity=*"center"*

android:inputType=*"textPersonName"* >

<requestFocus />

</EditText>

<Button

android:id=*"@+id/rsubmit"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:layout\_marginLeft=*"30dp"*

android:layout\_marginTop=*"50dp"*

android:layout\_marginRight=*"30dp"*

android:gravity=*"center"*

android:text=*"submit"* />

</LinearLayout>

**7.SYSTEM TESTING**

Software Testing is the process used to help identify the correctness, completeness, security, and quality of developed computer software. Testing is a process of technical investigation, performed on behalf of stakeholders, that is intended to reveal quality-related information about the product with respect to the context in which it is intended to operate. This includes, but is not limited to, the process of executing a program or application with the intent of finding errors. Quality is not an absolute; it is value to some person. With that in mind, testing can never completely establish the correctness of arbitrary computer software; testing furnishes a criticism or comparison that compares the state and behavior of the product against a specification. An important point is that software testing should be distinguished from the separate discipline of Software Quality Assurance (SQA), which encompasses all business process areas, not just testing.

There are many approaches to software testing, but effective testing of complex products is essentially a process of investigation, not merely a matter of creating and following routine procedure. One definition of testing is "the process of questioning a product in order to evaluate it", where the "questions" are operations the tester attempts to execute with the product, and the product answers with its behavior in reaction to the probing of the tester[citation needed]. Although most of the intellectual processes of testing are nearly identical to that of review or inspection, the word testing is connoted to mean the dynamic analysis of the product—putting the product through its paces. Some of the common quality attributes include capability, reliability, efficiency, portability, maintainability, compatibility and usability. A good test is sometimes described as one which reveals an error; however, more recent thinking suggests that a good test is one which reveals information of interest to someone who matters within the project community.

**7.1 Introduction:**

In general, software engineers distinguish software faults from software failures. In case of a failure, the software does not do what the user expects. A fault is a programming error that may or may not actually manifest as a failure. A fault can also be described as an error in the correctness of the semantic of a computer program. A fault will become a failure if the exact computation conditions are met, one of them being that the faulty portion of computer software executes on the CPU. A fault can also turn into a failure when the software is ported to a different hardware platform or a different compiler, or when the software gets extended. Software testing is the technical investigation of the product under test to provide stakeholders with quality related information. Software testing may be viewed as a sub-field of Software Quality Assurance but typically exists independently (and there may be no SQA areas in some companies). In SQA, software process specialists and auditors take a broader view on software and its development. They examine and change the software engineering process itself to reduce the amount of faults that end up in the code or deliver faster.

Regardless of the methods used or level of formality involved the desired result of testing is a level of confidence in the software so that the organization is confident that the software has an acceptable defect rate. What constitutes an acceptable defect rate depends on the nature of the software. An arcade video game designed to simulate flying an airplane would presumably have a much higher tolerance for defects than software used to control an actual airliner.

A problem with software testing is that the number of defects in a software product can be very large, and the number of configurations of the product larger still. Bugs that occur infrequently are difficult to find in testing. A rule of thumb is that a system that is expected to function without faults for a certain length of time must have already been tested for at least that length of time. This has severe consequences for projects to write long-lived reliable software.

A common practice of software testing is that it is performed by an independent group of testers after the functionality is developed but before it is shipped to the customer. This practice often results in the testing phase being used as project buffer to compensate for project delays. Another practice is to start software testing at the same moment the project starts and it is a continuous process until the project finishes.

Another common practice is for test suites to be developed during technical support escalation procedures. Such tests are then maintained in regression testing suites to ensure that future updates to the software don't repeat any of the known mistakes. It is commonly believed that the earlier a defect is found the cheaper it is to fix it. Unit tests are maintained along with the rest of the software source code and generally integrated into the build process (with inherently interactive tests being relegated to a partially manual build acceptance process).The software, tools, samples of data input and output, and configurations are all referred to collectively as a test harness.

**7.2 History:**

The separation of debugging from testing was initially introduced by Glen ford J. Myers in his 1978 book the "Art of Software Testing". Although his attention was on breakage testing it illustrated the desire of the software engineering community to separate fundamental development activities, such as debugging, from that of verification. Drs. Dave Gelperin and William C. Hetzel classified in 1988 the phases and goals in software testing as follows: until 1956 it was the debugging oriented period, where testing was often associated to debugging: there was no clear difference between testing and debugging. From 1957-1978 there was the demonstration oriented period where debugging and testing was distinguished now - in this period it was shown, that software satisfies the requirements.

From 1988 on it was seen as prevention oriented period where tests were to demonstrate that software satisfies its specification, to detect faults and to prevent faults. Dr. Gelperin chaired the IEEE 829-1988 (Test Documentation Standard) with Dr. Hetzel writing the book "The Complete Guide of Software Testing". Both works were pivotal in to today's testing culture and remain a consistent source of reference. Dr. Gelperin and Jerry E. Durant also went on to develop High Impact Inspection Technology that builds upon traditional Inspections but utilizes a test driven additive.

* **Testing Methodologies**
* Black box Testing:
* White box Testing.
* Gray Box Testing.
* **Levels of Testing**
  + Unit Testing.
  + Module Testing.
  + Integration Testing.
  + System Testing.
  + User Acceptance Testing
* **Types Of Testing**
  + Smoke Testing.
  + Sanitary Testing.
  + Regression Testing.
  + Re-Testing.
  + Static Testing.
  + Dynamic Testing.
  + Alpha-Testing.
  + Beta-Testing.
  + Monkey Testing.
  + Compatibility

**TCD (Test Case Documentation)**

* **STLC**
  + Test Planning.
  + Test Development.
  + Test Execution.
  + Result Analysis.
  + Bug-Tracing.
  + Reporting.
* **Microsoft Windows – Standards**
* **Manual Testing**
* **Automation Testing (Tools)**
  + Win Runner.
  + Test Director.

**7.3 Testing:**

* The process of executing a system with the intent of finding an error.
* Testing is defined as the process in which defects are identified, isolated, subjected for rectification and ensured that product is defect free in order to produce the quality product and hence customer satisfaction.

**7.3.1 Testing Methodologies:**

* **Black box Testing**: is the testing process in which tester can perform testing on an application without having any internal structural knowledge of application.

Usually Test Engineers are involved in the black box testing

* **White box Testing**: is the testing process in which tester can perform testing on an application with having internal structural knowledge.

Usually The Developers are involved in white box testing..

* **Gray Box Testing**: is the process in which the combination of black box and white box tonics’ are used.

**7.3.2 Levels of Testing:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | ***Module1Module2Module3***   |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | ***Units***   |  |  |  | | --- | --- | --- | |  |  |  | |  | ***Units***   |  |  |  | | --- | --- | --- | |  |  |  | |  | ***Units***   |  |  |  | | --- | --- | --- | |  |  |  | |   i/p ***Integration***  o/p i/p ***Integration o/p*** |   ***System Testing: Presentation + business +Databases***  ***🚹UAT: user acceptance testing*** |

**7.3.3 Test Planning:**

**1.**Test Plan is defined as a strategic document which describes the procedure how to perform various testing on the total application in the most efficient way.

**2.**This document involves the scope of testing,

**3.** Objective of testing,

**4.** Areas that need to be tested,

**5.** Areas that should not be tested,

**6.** Scheduling Resource Planning,

**7.** Areas to be automated, various testing tools used.

**Test Development**:

**1.** Test case Development (check list)

**2.** Test Procedure preparation. (Description of the Test cases).

**Result Analysis**:

**1.** Expected value: is nothing but expected behavior of application.

**2.** Actual value: is nothing but actual behavior ofApplication

**Bug Tracing:** Collect all the failed cases, prepare documents.

**Reporting:** Prepare document (status of the application)

**Types Of Testing:**

**🚺>Smoke Testing**: is the process of initial testing in which tester looks for the availability of all the functionality of the application in order to perform detailed testing on them. (Main check is for available forms)

**🚺>Sanity Testing:** is a type of testing that is conducted on an application initially to check for the proper behavior of an application that is to check all the functionality are available before the detailed testing is conducted by on them.

**🚺>Regression Testing:** is one of the best and important testing. Regression testing is the process in which the functionality, which is already tested before, is once again tested whenever some new change is added in order to check whether the existing functionality remains same.

**🚺>Re-Testing:** is the process in which testing is performed on some functionality which is already tested before to make sure that the defects are reproducible and to rule out the environments issues if at all any defects are there.

**🚺Static Testing:** is the testing, which is performed on an application when it is not been executed.ex: GUI, Document Testing

**🚺Dynamic Testing:** is the testing which is performed on an application when it is being executed.ex: Functional testing.

**🚺Alpha Testing:** it is a type of user acceptance testing, which is conducted on an application when it is just before released to the customer.

**🚺 Beta-Testing:** it is a type of UAT that is conducted on an application when it is released to the customer, when deployed in to the real time environment and being accessed by the real time users.

**🚺Compatibility testing:** it is the testing process in which usually the products are tested on the environments with different combinations of databases

**🚺Installation Testing:** it is the process of testing in which the tester try to install or try to deploy the module into the corresponding environment by following the guidelines produced in the deployment document and check whether the installation is successful or not.

**🚺Adhoc Testing:** Adhoc Testing is the process of testing in which unlike the formal testing where in test case document is used, with out that test case document testing can be done of an application, to cover that testing of the future which are not covered in that test case document. Also it is intended to perform GUI testing which may involve the cosmotic issues.

**Test Case Document Contains :**

* Test Scope (or) Test objective
* Test Scenario
* Test Procedure
* Test case

This is the sample test case document for the Acadamic details of student project:

**Test scope:**

Test coverage is provided for the screen “ Acadamic status entry” form of a student module of university management system application.

**Test Scenario:**

When the office personals use this screen for the marks entry, calculate the status details, saving the information on student’s basis and quit the form.

**Test Procedure:**

The procedure for testing this screen is planned in such a way that the data entry, status calculation functionality, saving and quitting operations are tested in terms of Gui testing, Positive testing, Negative testing using the corresponding Gui test cases, Positive test cases, Negative test cases respectively.

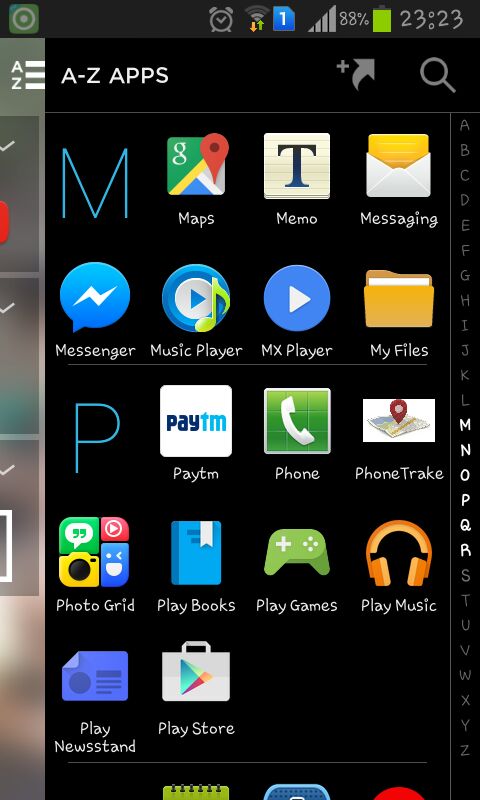
**7.4 Test Cases:**

Template for Test Case

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| T.C.No | Description | Exp | Act | Result |
| 1. | Installation of the app | True/False | True | Successful installation of the app |
| 2. | Clicking on the app icon | True/False | True | Entered into the app. |
| 3. | Register | True/False | True | If the register is completed it navigates to other page |
| 4. | Giving inputs in the text fields for “name” and “mobile number” | True/False | True | If both the inputs are acceptable |
| 5. | Giving inputs in the text fields for “name” and “mobile number” | True/False | False | If both the inputs are null values |
| 6. | Entering register button after successful registeration | True/False | True | The user navigates to the home/main page. |
| 7. | Accessing the main page | True/False | True | User finds three buttons,”location”,  ”tracking”,Sms” |
| 8. | Clicking on the “location” button | True/False | True | User can get the longitude and latitude values. |
| 9. | Clicking on the “tracking” button | True/False | True | User can get the specified location by google maps. |
| 10. | Clicking on the “send sms” | True/ False | True | The tracking information can be sent to the specified user. |

**8. SCREEN SHOTS**

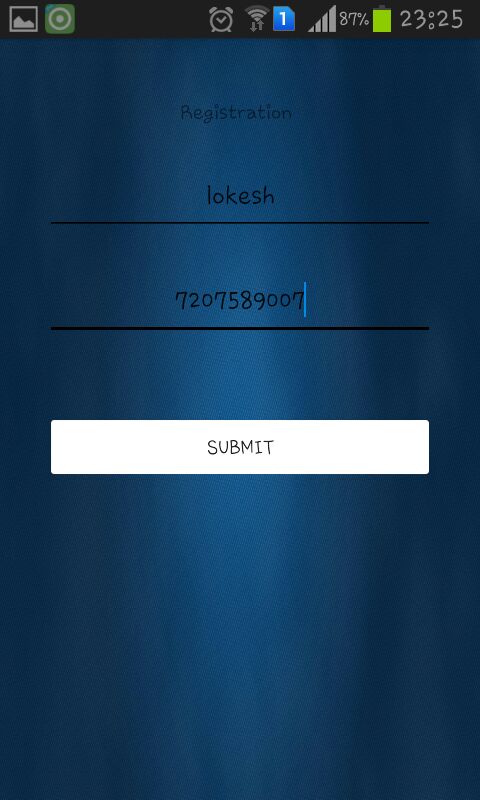
**1.Menu page**

****

**Fig:8.1 Menu page**

* The above screenshot is after installing the mobile app in a respective user mobile.

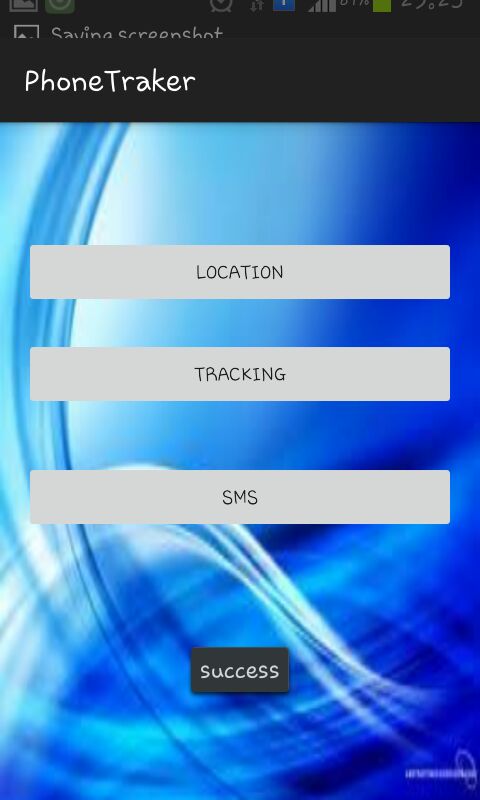
2.Registration page



**Fig:8.2 Registration page**

* The above screenshot illustrates the registration form which consists of name and number of parent for which we should send the tracking details.
* By clicking on the submit button it navigates to other page.

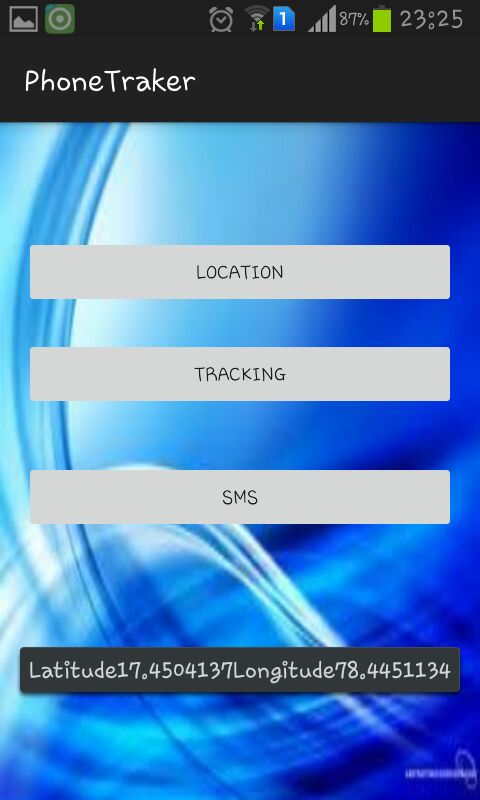
3.home page



**Fig 8.3 home page**

This screenshot illustrates that the successful registration and navigated to the homepage which consists of 3 buttons location, tracking and sms.

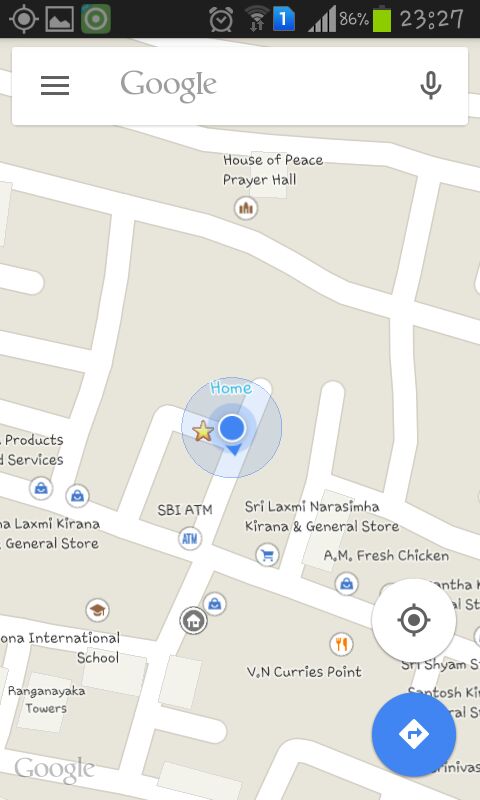
4.location page

****

**Fig 8.4 location page**

When we click on the location button we will get the present location latitude and longitude values which can be shown in above screenshot.

5.Tracking location

****

**fig:8.5 tracking location**

The above screenshot illustrates when user click on the tracking button in the home page ,this will navigate to the new page where we can see the google maps, the current location of the user will be shown in the map.

6.SMS sending

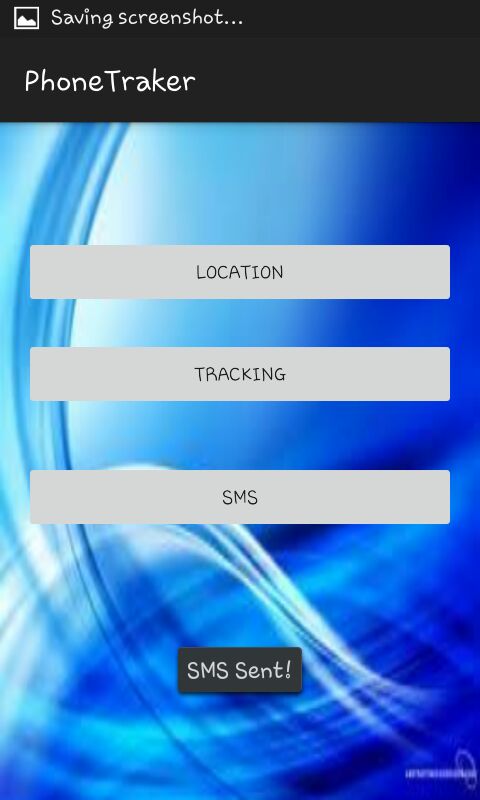


Fig:8.6sms sending

The above screenshot is illustrated when user clicks on sms button it automatically sends the message to the specified number.

\

7.receiving msg

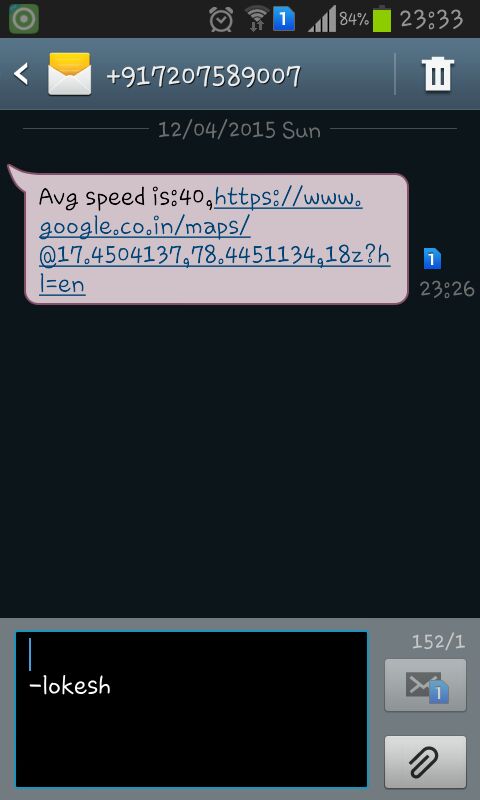


Fig: 8.7 receiving msg

This screenshot explains the parent side receiving the message with url of location and speed.

8.Opening URL

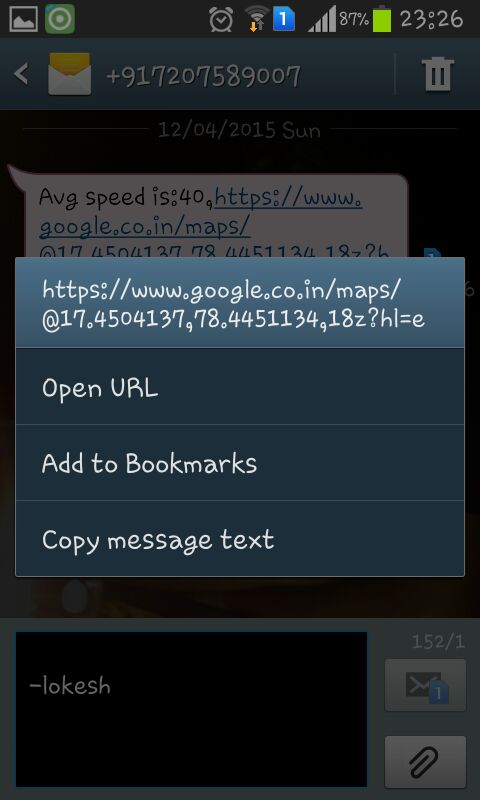


Fig.8.8 opening url

By clicking the url in the parent message we will get the above options as shown in the screenshot.

9.Mode of action

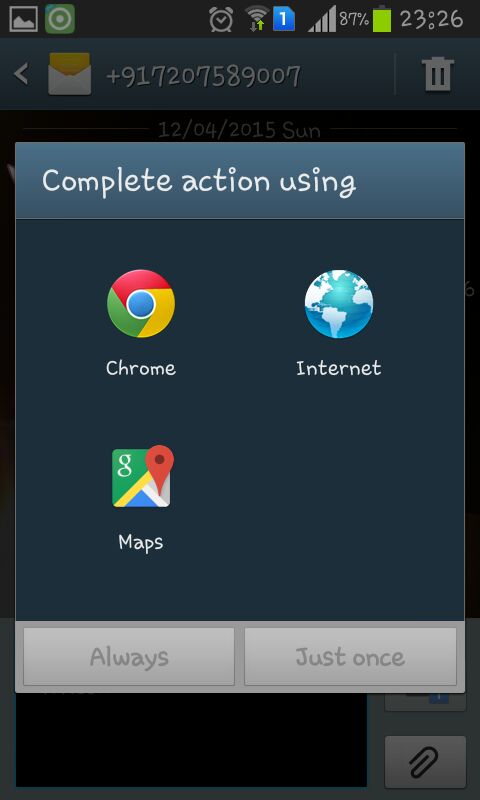


Fig:8.9mode of action

By selecting the open url option we will get the following options as shown in the screenshot.

10.Select google maps

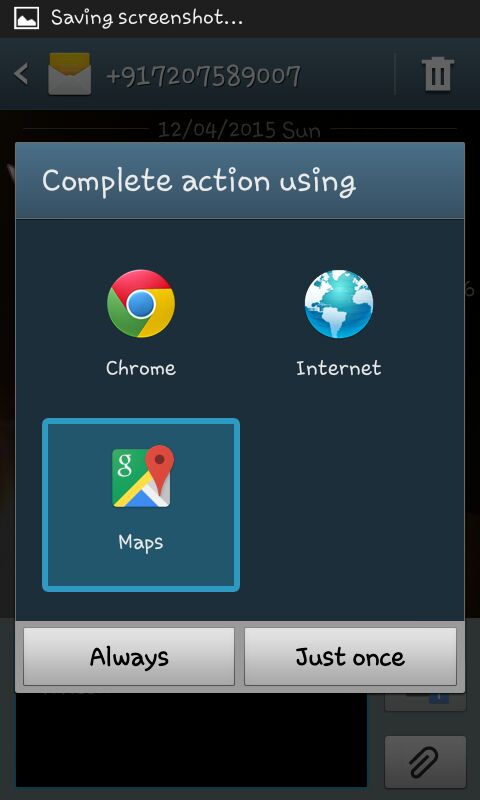


Fig:8.10 selecting google maps

The url can be opened by using the maps to see the current location of the user.

11.loacting in maps

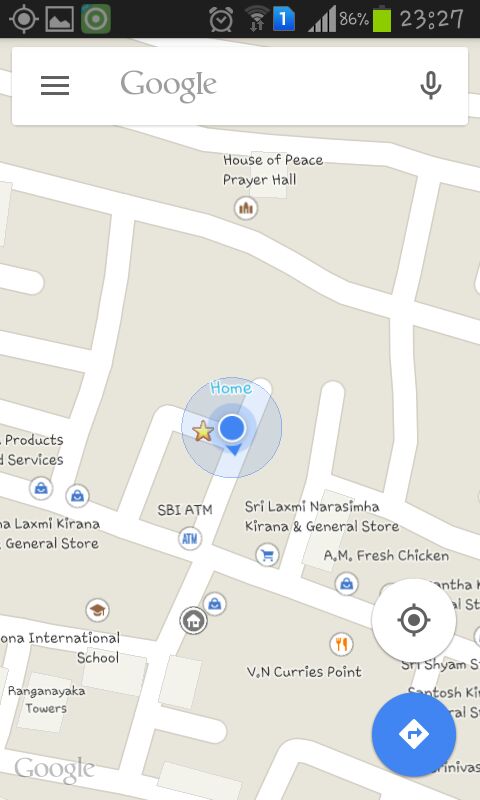


Fig 8.11 location in maps

The current location is tracked and can be seen .

**9. LIMITATIONS AND SCOPE FOR FUTURE**

**ENHANCEMENTS**

**9.1 Limitations of the system:**.

* + Application works only when it is connected to internet.
  + The average speed is notified.

**9.2 Future Enhancements:**

It is not possible to develop a system that makes all the requirements of the user. User requirements keep changing as the system is being used. Some of the future enhancements that can be done to this system are:

* As the technology emerges, it is possible to upgrade the system and can be adaptable to desired environment.
* Because it is based on object-oriented design, any further changes can be easily adaptable.
* Based on the future security issues, security can be improved using emerging technologies.
* Since it is an mobile application based on android, the android versions are being upgraded day to day, so this app should be able to configure with any version of android.

## 10.CONCLUSION

**10.1 PROJECT SUMMARY:**

This application software has been computed successfully and was also tested successfully by taking “test cases”. It is user friendly, and has required options, which can be utilized by the user to perform the desired operations.

The software is developed using as front end ECLIPSE and SQLite as back end in Windows environment.

The goals that are achieved by the software are:

* Access to the mobile application.
* Efficient management of sending/retrieving messages.
* Secured authentication, personal to the specific user.
* User friendly.
* Portable and flexible for further enhancement.

**WORK DONE:**

This project involves an Android application which communicates messages to the other user on button click . This project offers user the facility to register using which he can authenticate himself and then access the app. Application is flexible to be used by any new user, as it provides details about the app and its working.

**GOALS**

* Reduced human effort
* Easy to track the location.
* Accessing the location easily by using GPS system.
* Notifying the location easily by sending the URL to the specified user.

**BIBILIOGRAPHY**

* <http://developer.android.com/guide/topics/connectivity/wifip2p.html>
* <http://developer.android.com/training/connect-devices-wirelessly/nsd.html>
* <http://developer.android.com/training/connect-devices-wirelessly/wifi-direct.html>
* [www.youtube.com/watch?v=nBqZ6lQ3IVQ](http://www.youtube.com/watch?v=nBqZ6lQ3IVQ)
* <http://www.safetrax.in/?gclid=CjwKEAiArqKmBRCOj_qfmuqinnYSJAAkAYwGDrEKQCKuXOq5-YFR6LIoz9OsS7I_frT2vuYHCIb8xBoCmCLw_wcB>
* <https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0CCwQFjAA&url=http%3A%2F%2Fwww.vehiclelocationsystem.com%2F&ei=0rbJVI6kJuK3mwXn64DgAQ&usg=AFQjCNHZ_qx8JL9E9aobCSopkXb8S0ys6g&sig2=Gl0oajwZPC0QQTdkcvfEzA>