**CHAPTER 1**

**INTRODUCTION**

* 1. **OVERVIEW OF THE PROJECT**

In current system, in order to find out the location of friends, user need to call his/her friend about their location.The proposed system will helps the user to find out friends location on a Google map as well as calculates the distance between both the users

.

Everything is possible in LBS. The term LBS refers to an IT service which provides information that has been filtered , selected , compiled, or created, taking into account the current locations of the device, other people, or mobile objects. Mobile commerce has evolved to utilize end user’s location data to deliver relevant, timely, and engaging content. For mobile network operators, Location Based Services represent a welcome additional revenue stream, which can be generated by leveraging their current on the investments in fixed infrastructure.

LBSs are attractive as consumers do not need to enter location to the information manually, but instead are automatically pinpointed and tracked. A location - based service on a small device usually initially provides only small portions of information to the user who then has the opportunity to access further data as required. The application “ Friend Locator on Mobiles ” solves all of these are problems. It offers below services

1. Allows user to select friend for his location updates.
2. Uploads user’s current location at specific frequency.
3. Get friends current location
4. Shows friends location on google Map.
5. Shows distance in kilometers for friends away from user.

**1.2 LITERATURE SURVEY**

The need to find some ones location is enormous now a days. Especially in a friends circle or in a family it is a necessity to know the location now and then. Already existing systems are little efficient. Everytime calling and asking the location to someone is little cumbersome, when the number of times to ask is more. Also instead of simply knowing the locations, it would be impressive to see the exact place by using map. All the android mobile

Devices are available with Global Positioning System in which Location based services can be established. Some system is certainly required not only to reduce the number of calls made and eliminating the costs, but also to use the already available technologies in android in a efficient manner. With the advancement of internet facility, even the mobile devices can have the data connectivity to a good extent of strength. With the increasing number of android devices around, the growth towards the technology may further enhance the Location Based Services.

**1.3 EXISTING SYSTEM**

* In current system, in order to find out a location of friends, user needs to call and ask friend about his/her location.
* The existing location identifiers find out the approximate location, but not the exact location.

Also The RF based location identifiers are limited only to a particular range. And in GSM technique, the accuracy is poor.

**1.4 PROPOSED SYSTEM**

The Usual GSM technique is replaced by Global Positioning System (GPS)Technique which helps to find the location in a greater accuracy. WCF (Windows communication Foundation) and REST ( Representational State Transfer protocol) methods are implemented. This a Client-Server System allows us to track the location by keeping a database in server in which latitude and longitude of all users are updated very often and whenever the other user wants to check the location, this information is sent and and the result is seen in google maps as well.

**ADVANTAGES**

* Here the accuracy is improver to a greater extent.
* Android map view helps to visualize the location instead of just knowing the location information.

**TECHNOLOGY**

Technologies used are

* Android Google API.
* GPS (for getting user’s current location)
* GPRS (for internet connectivity between mobile and server)

**ALGORITHM/METHODOLOGY:**

WCF (Windows communication Foundation) REST service.

REST : (REpresentational State Transfer protocol)

**1.5 LBS TECHNOLOGY OVERVIEW**

Location-based services are a general class of computer program-level services used to include specific controls for location and time data as control features in computer programs. As such (LBS) is an information and has a number of uses in Social Networking today as an entertainment service, which is accessible with [mobile devices](http://en.wikipedia.org/wiki/Mobile_device) through the [mobile network](http://en.wikipedia.org/wiki/Mobile_network) and which uses information on the geographical position of the mobile device. This has become more and more important with the expansion of the smart phone and tablet markets as well.

LBS are used in a variety of contexts, such as health, indoor object search, entertainment work, personal life, etc

LBS include services to identify a location of a person or object, such as discovering the nearest banking cash machine (a.k.a. [ATM](http://en.wikipedia.org/wiki/Automated_teller_machine)) or the whereabouts of a friend or employee. LBS include parcel tracking and [vehicle tracking](http://en.wikipedia.org/wiki/Vehicle_tracking) services. LBS can include [mobile commerce](http://en.wikipedia.org/wiki/Mobile_commerce) when taking the form of coupons or advertising directed at customers based on their current location. They include personalized weather services and even location-based games. They are an example of [telecommunication convergence](http://en.wikipedia.org/wiki/Telecommunication_convergence).

**1.6 GPS TECHNOLOGY OVERVIEW**

The Global Positioning System (GPS) is a space-based [satellite navigation](http://en.wikipedia.org/wiki/Satellite_navigation) system that provides location and time information in all weather conditions, anywhere on or near the Earth where there is an unobstructed line of sight to four or more GPS satellites. The system provides critical capabilities to military, civil and commercial users around the world. It is maintained by the United States government and is freely accessible to anyone with a [GPS receiver](http://en.wikipedia.org/wiki/GPS_receiver).

The GPS project was developed in 1973 to overcome the limitations of previous navigation systems,integrating ideas from several predecessors, including a number of classified engineering design studies from the 1960s. GPS was created and realized by the [U.S. Department of Defense](http://en.wikipedia.org/wiki/U.S._Department_of_Defense) (DoD) and was originally run with 24 satellites. It became fully operational in 1994. [Roger L. Easton](http://en.wikipedia.org/wiki/Roger_L._Easton) is generally credited as its inventor.

**CHAPTER 2**

**SYSTEM ANALYSIS**

**2.1 INTRODUCTION**

This project is mainly used for finding the location using GPS instead of using the service provider. With the advanced data communication in mobile devices, this project will work very efficiently.

**2.2 OVERALL DESCRIPTION**

In the friend mapper using LBS, the requirements are mobile phones that has wireless data connectivity and Global Positioning System(GPS).

GPS track the exact location at any instant of time and the final requirement is the internet connectivity in mobile phones.

**2.2.1 Product Description**

Friend Mapper using GPS involves finding the exact location of friends, family members or employees of a company. The Clients should have mobile phones having GPS. The Mobile phones should be connected to Internet as well as the server. All the users are given a username and password. Any user can track other users at a particular time.

**2.2.2 Product Functions**

Our Approach adopts wireless communication, does not requires wires. Our Approach used mobile instead of pc. It means we provide mobility. Also it helps making use of mobile services. These are the advantages of this method.

**2.2.3 Product Perspective**

The Project includes the following stages.

* Signup
* Choose the person
* View Location through Map
* Sign out

**SIGNUP**

There is a quality signup window because this is more secure than normal signup window. The signup window has text boxes in which user has to give his mobile number as the username and password where the location information, latitude and longitude is get automatically.

**CHOOSE THE PERSON**

Once the signup is successful the user can view other users location.

For that the user has to enter his/her mobile number and password along with the mobile number of whom’s location we are about to find out.

**VIEW LOCATION THROUGH MAP**

When the user enters above information correctly, the application takes the user to a map and it shows the location of the person whom we want to find as well as our own location. The accuracy purely depends on the strength of the data connection in mobile phones.

**SIGNOUT**

Furthur, the user can view other available users location or he/she can simply signout of the application using the available options.

**2.2.4 Requirement Specification :**

The requirements specification is a technical specification of the all requirements for the software products. It is the first step in requirements analysis process it lists the requirements of a particular software system including functional, performance and security requirements. These all the requirements also provide usage scenarios from a user, an operational and an administrative perspective. The purpose of software requirements to all the specification is to provide a detailed overview of the software project, its parameters and goals. This describes the project target audience and its user interface, hardware and software requirements.

**2.2.5 Operating Environment**

**HARDWARE REQUIREMENTS**

Mobile which has below features (The app can also run on emulator)

* GPS
* GPRS
* Android phone

**SOFTWARE REQUIREMENTS**

* Android SDK 1.5 or above.
* JDK1.6
* Eclipse IDE
* MySql Server5.5
* SQLyog

**2.3 SOFTWARE DESCRIPTION**

**2.3.1 JAVA PROGRAMMING LANGUAGE**

### Java programming language is a high-level language that can be characterized by all of the following buzzwords:

* Simple
* Object oriented
* Portable
* Distributed
* High performance
* Interpreted
* Multithreaded
* Robust
* Dynamic
* Secure

With most programming languages, we either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first we translate a program into an intermediate language called Java byte codes —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.



Fig 2.1 Compiler and Interpreter

We can think of Java byte codes as the machine code instructions for the Java Virtual Machine *(*Java VM). Every Java interpreter, whether it’s a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java byte codes help make “write once, run anywhere” possible. We can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.



Fig 2.2 Java Execution

### 2.3.2 JAVA PLATFORM

**Platform** is the hardware or software environment in which a program runs. Some of the most popular platforms like Windows 2000, Linux, Solaris, and MacOScanare described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it’s a software-only platform that runs on top of other hardware-based platforms.

The Java platform has two components:

* The Java Virtual Machine (Java VM)
* The Java Application Programming Interface (Java API)

The Java VM is the base for the Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as packages. The following figure depicts a program that is running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.



Fig 2.3 Java Platform

Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability.

**2.3.3 FEATURES OF JAVA PLATFORM**

* **Essentials**: Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.
* **Applets**: The set of conventions used by applets.
* **Networking**: URLs, TCP (Transmission Control Protocol), UDP (User Data gram Protocol) sockets, and IP (Internet Protocol) addresses.
* **Internationalization**: Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.
* **Software components**: Known as JavaBeans, can plug into existing component architectures.
* **Object serialization**: Allows lightweight persistence and communication via Remote Method Invocation (RMI).
* **Java Database Connectivity (JDBC)**: Provides uniform access to a wide range of relational databases. The Java platform also has APIs for 2D and 3D graphics, accessibility, servers, collaboration, telephony, speech, animation, and more. The following figure depicts what is included in the Java 2 SDK.



Fig 2.4 Java 2 SDK

**2.3.4 JSP**

**Java Server Pages (JSP)** is a technology that helps software developers to create dynamically generated web pages based on HTML, XML or other document types.To deploy and run JSP, a compatible web server with a servlet container such as Apache Tomcat or Jetty server is required.

JSP may be viewed as a high-level [abstraction](http://en.wikipedia.org/wiki/Abstraction_(computer_science)) of [Java servlets](http://en.wikipedia.org/wiki/Java_servlet). JSPs are translated into [servlets](http://en.wikipedia.org/wiki/Java_Servlet) at runtime; each JSP's servlet is cached and re-used until the original JSP is modified.

JSP can be used independently or as the view component of a server-side [model–view–controller](http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller) design, normally with [JavaBeans](http://en.wikipedia.org/wiki/JavaBeans) as the model and Java servlets (or a framework such as [Apache Struts](http://en.wikipedia.org/wiki/Apache_Struts)) as the controller. This is a type of [Model 2](http://en.wikipedia.org/wiki/Model_2) architecture.

JSP allows Java code and certain pre-defined actions to be interleaved with static web markup content, with the resulting page being compiled and executed on the server to deliver a document. The compiled pages, as well as any dependent Java libraries, use Java byte code rather than a native software format. Like any other Java program, they must be executed within a [Java virtual machine](http://en.wikipedia.org/wiki/Java_virtual_machine) (JVM) that integrates with the server's host operating to provide an abstract platform-neutral environment.

JSPs are usually used to deliver HTML and XML documents, but through the use of Output Stream, they can deliver other types of data as well. The [Web container](http://en.wikipedia.org/wiki/Web_container) creates JSP implicit objects like pageContext, servletContext, session, request & response.

**SYNTAX**

JSP pages use several delimiters for scripting functions. The most basic is <% ... %>, which encloses a JSP scriptlet. A scriptlet is a fragment of Java code that is run when the user requests the page. Other common delimiters include <%= ... %> for expressions, where the value of the expression is placed into the page delivered to the user, and directives, denoted with <%@ ... %>.

Java code is not required to be complete or self-contained within its scriptlet element block, but can straddle markup content providing the page as a whole is syntactically correct. For example, any Java if/for/while blocks opened in one scriptlet element must be correctly closed in a later element for the page to successfully compile. Markup which falls inside a split block of code is subject to that code, so markup inside an if block will only appear in the output when the if condition evaluates to true; likewise, markup inside a loop construct may appear multiple times in the output depending upon how many times the loop body runs.

**2.3.5 APACHE TOMCAT SERVER**

Apache Tomcat (formerly under the Apache Jakarta Project. Tomcat is now a top level project) is a web container developed at the Apache Software Foundation. Tomcat implements the servlet and the Java Server Pages ([JSP](http://www.codebeach.com/tutorials/what-is-jsp.asp)) specifications from Sun Microsystems, providing an environment for Java code to run in cooperation with a web server. It adds tools for configuration and management but can also be configured by editing configuration files that are normally [XML](http://www.codebeach.com/tutorials/what-is-xml.asp)-formatted. Because Tomcat includes its own HTTP server internally, it is also considered as a standalone web server.

Tomcat is a web server that supports servlets and JSPs. Tomcat comes with the Jasper compiler that compiles JSPs into servlets. The Tomcat servlet engine is often used in combination with an Apache web server or other web servers. Tomcat can also function as an independent web server. Earlier in its development, the perception existed that standalone Tomcat was only suitable for development environments and other environments with minimal requirements for speed and transaction handling. However, that perception no longer exists; Tomcat is increasingly used as a standalone web server in high-traffic, high-availability environments.

Since its developers wrote Tomcat in Java, it runs on any operating system that has a JVM.

**PRODUCT FEATURES**

Tomcat 3.x (initial release)

* implements the Servlet 2.2 and JSP 1.1 specifications
* Servlet reloading
* Basic http functionality tomcat 4.x
* Implements the servlet 2.3 and JSP 1.2 specifications
* Servlet container redesigned as Catalina
* JSP engine redesigned as jasper
* Coyote connector
* Java management extensions (JMX), JSP and struts-based administration
* Tomcat 5.x
* Implements the servlet 2.4 and JSP 2.0 specifications
* Reduced garbage collection, improved performance and scalability
* Native windows and Unix wrappers for platform integration
* Faster JSP paring

**2.3.6 DATABASE CONNECTIVITY**

**2.3.6.1 JDBC**

In an effort to set an independent database standard API for Java; Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC offers a generic SQL database access mechanism that provides a consistent interface to a variety of RDBMSs. This consistent interface is achieved through the use of “plug-in” database connectivity modules, or drivers. If a database vendor wishes to have JDBC support, he or she must provide the driver for each platform that the database and Java run on.

To gain a wider acceptance of JDBC, Sun based JDBC’s framework on ODBC. As you discovered earlier in this chapter, ODBC has widespread support on a variety of platforms. Basing JDBC on ODBC will allow vendors to bring JDBC drivers to market much faster than developing a completely new connectivity solution.

JDBC was announced in March of 1996. It was released for a 90 day public review that ended June 8, 1996. Because of user input, the final JDBC v1.0 specification was released soon after.

**2.3.7 INTRODUCTION TO ANDROID**

Android is the software platform from Google and the Open Handset Alliance that some say has the potential to revolutionize the global cell phone market. Android is a software environment built for mobile devices. It is not a hardware platform. While components of the underlying OS are written in C or C++, user applications are built for Android in Java.

In the Android platform, there is no difference between the built-in applications and applications created with the SDK. This means that powerful applications can be written to tap into the resources available on the device.

**LIBRARIES**

Android includes a set of C/C++ libraries used by various components of the Android system. These capabilities are exposed to developers through the Android application framework. Some of the core libraries are listed below:

**System C library** - a BSD-derived implementation of the standard C system library (lib), tuned for embedded Linux-based devices

**Media Libraries** - based on Packet Video’s Open CORE; the libraries support playback and recording of many popular audio and video formats, as well as static image files, including MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG

**Surface Manager** - manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications

**LibWebCore**- a modern web browser engine which powers both the Android browser and an embeddable web view

**SGL** - the underlying 2D graphics engine

**3D libraries** - an implementation based on OpenGL ES 1.0 APIs; the libraries use either hardware 3D acceleration (where available) or the included, highly optimized 3D software rasterizer

**Free Type** - bitmap and vector font rendering

**SQLite** - a powerful and lightweight relational database engine available to all applications.

**ANDROID RUNTIME**

Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language.

Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently. The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint. The VM is register-based, and runs classes compiled by a Java language compiler that have been transformed into the .dex format by the included "dx" tool.

The Dalvik VM relies on the Linux kernel for underlying functionality such as threading and low-level memory management.

**2.4 SYSTEM ARCHITECTURE**

The Architecture explain the process involved in the system. The architecture of our project picturizes that the SQL Server contains a database. When the user signups with his/her information, all these data has been transferred to the database available in the server. From then on the Location information is continuously updated to the server. When the user tries to obtain others location, the data is fetched from the database and shown in map. The change in location is continuously updated until the data connectivity strength is good.

View Map

Request location

Signup

Database

Fig 2.5 System Architecture Diagram

**2.5 SOFTWARE QUALITY ATTRIBUTE**

The required functions are available, including interoperability and security.

**Reliability:**

It is the ability of system to recover from the failure after detection.

This project is prone to fault tolerance and recoverability.

**Reusability:**

It is the ability of the system to be reused. This project can be reused in various domains.

**Usability:**

It is effectiveness of the use of system. The user interface of this project is developed in such a way that the user can understand easily, learn at a faster rate and operate with great level of comfort.

**Effeciency:**

This project is having high performance and resource behavior.

**Portability:**

It is ability of system to adapt different environment. This project is platform independent, it can be transferred and installed in different operating environments, hence it is portable.

**Integrity:**

It is the protection of the system from unauthorized access. This project provides a higher degree of integrity.

**CHAPTER 3**

**DETAILED DESIGN AND TEST PLAN**

**3.1 DECOMPOSTION DESCRIPTION**

There are various components and modules present in our project. Everything in the project can’t be done at the same time; hence we had decomposed our project into some of the modules. Decomposition gives us an easier way to solve any type of problem. It has a power to convert very huge things into number of small ones. Therefore the work stress will be reduced and much possibility to understand the project very easily.

**3.1.1 Module Decomposition**

Module is the part of project. Module decomposition is that dividing the work into smaller parts such a way that to make our work easier to understand and face the problems that occurs while handling the project work. So based upon the project the module and decomposition had been done.

**3.2 DETAILED DESIGN**

In our project the modules had been designed based upon its work. The design of our project is that to build a new architecture for the Location Based Services.

**3.3 OOAD DIAGRAMS**

The typical Object oriented analysis and design diagrams for the project description can be shown according to the work we have done on the project. The OOAD diagrams consist of following categories.

**3.4.1 USE CASE DIAGRAM**

A Use Case diagram is a graph of actors, a set of use cases enclosed by a system boundary, a communication association between the actors and the use cases, and a generalization among use cases.

****

Fig 3.1 Use-case Diagram

Figure 3.1 represents use case diagram. The actors in the use case diagram are the user and server. A user Signups the location, requests to view a location of another user, communicating with the server and obtains the location and view it in map.

**3.4.2 ACTIVITY DIAGRAM**

An activity diagram is a variation or special case of a state machine, in which the states are the activities representing the performance of operations and the transitions are triggered by the completion of operation.

****

Fig 3.2 Activity Diagram

Figure 3.2 repersents the activity diagram. It gives the step by step explanation of the activities. Once a user registers and logs in, username and password are checked for authentication. Then the user is allowed to view the other registered users location.

**3.4.3 CLASS DIAGRAM**

A class diagram in the [Unified Modeling Language](http://en.wikipedia.org/wiki/Unified_Modeling_Language) (UML) is a type of static structure diagram that describes the structure of a system by showing the system's [classes](http://en.wikipedia.org/wiki/Class_(computer_science)), their attributes, operations (or methods), and the relationships among the classes.

****

Figure 3.3 Class Diagram

Figure 3.3 represents the class diagrams. In this diagram relationships between the classes reistration, login member view and server are establishes and their attributes and methods are defined.

**3.4.4 COMPONENT DIAGRAM**

The collaboration diagram represents the collaboration, which is a set of objects related in a particular context, and interaction, which is othe set of messages exchanged among the objects within the collaboration to achieve the desired outcome.

****

Fig 3.4 Collaboration diagram

Figure 3.4 represents the collaboratin diagram. Here the mobile user and the application server are the components. Registration, login , member view request, latitude longitude value are the messages exchanged between the components.

**3.4.5 SEQUENCE DIAGRAM**

A sequence diagram describes the behaviour of the system by viewing the interaction between the system and the environment.

****

Figure 3.5 Sequence Diagram

Figure 3.3 represents the sequence diagram. It shows the sequence of events occurred. Here the request is made by user and the response is provided by the application server.

**TESTING**

The project is tested to verify its correctness and efficiency. The test plan includes following test.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.NO** | **UI ID** | **FIELD** | **INPUT** | **OUTPUT EXPECTED** | **OUTPUT OBTAINED** |
| 1. | SIGNUP ID | MOBILE NUMBER | NUMERICAL VALUE | ACCEPT THE DATA | ACCEPT THE DATA |
|  |  | PASSWORD | TEXT | ACCEPT THE DATA | ACCEPT THE DATA |
|  |  | STATIC IP(SERVER) | NUMERICAL VALUE | ACCEPT THE DATA | ACCEPT THE DATA |
|  |  | LATITUDE | NUMERICAL VALUE | GET DATA | GET DATA |
|  |  | LONGITUDE | NUMERICAL VALUE | GET DATA | GET DATA |

**Table 3.1 Test Plan for Signup form**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.NO** | **UI ID** | **FIELD** | **INPUT** | **OUTPUT EXPECTED** | **OUTPUT OBTAINED** |
| 1 | REQUEST FRIEND  LOCATION | MOBILE NUMBER | NUMERICAL VALUE | ACCEPT THE DATA | ACCEPT THE DATA |
|  |  | PASSWORD | TEXT | ACCEPT THE DATA | ACCEPT THE DATA |
|  |  | SECOND PERSON MOBILE NUMBER | NUMERICAL VALUE | ACCEPT THE DATA | ACCEPT THE DATA |
|  |  | STATIC IP ADDRESS | NUMERICAL VALUE | ACCEPT THE DATA | ACCEPT THE DATA |

**Table 3.2 Test Plan for Request Location**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.NO** | **UI ID** | **FIELD** | **INPUT** | **OUTPUT EXPECTED** | **OUTPUT OBTAINED** |
| 1 | VIEW GOOGLE MAP WITH LOCATION | NUMERICAL VALUES | LATITUDE | POINT IT IN MAP | POINT IT IN MAP |
|  |  | NUMERICAL VALUES | LONGITUDE | POINT IT IN MAP | POINT IT IN MAP |
|  |  | MAP VIEW | GOOGLE API | SHOW MAP | SHOW MAP |

**Table 3.3 Test Map for Map Viewing**

**3.6 FEASIBILITY REPORT**

3.6.1 **ECONOMIC FEASIBILITY**

Economic analysis is most frequently used for evaluation of the effectiveness of the system. More commonly known as cost/benefit analysis the procedure is to determine the benefit and saving that are expected from a system and compare them with costs, decisions is made to design and implement the system.

This part of feasibility study gives the top management the economic justification for the new system. This is an important input to the management the management, because very often the top management does not like to get confounded by the various technicalities that bound to be associated with a project of this kind. In the system, the organization is most satisfied by economic feasibility. Because, if the organization implements this system, it need not require any additional hardware resources as well as it will be saving lot of time.

**3.6.2 TECHNICAL FEASIBILITY**

Technical feasibility centres on the existing manual system of the test management process and to what extent it can support the system. According to feasibility analysis procedure the technical feasibility of the system is analyzed and the technical requirements such as software facilities, procedure, inputs are identified. It is also one of the important phases of the system development activities.

The system offers greater levels of user friendliness combined with greater Processing speed. Therefore, the cost of maintenance can be reduced. Since, processing speed is very high and the work is reduced in the maintenance point of view management convince that the project is operationally feasible.

**CHAPTER 4**

**IMPLEMENTATION AND RESULTS**

**4.1 MODULES**

* User Interface design
* GPS connectivity: (sync up of location)
* Server application
* Database connectivity for clients

**4.2 MODULE DESCRIPTION**

**Module 1: USER INTERFACE DESIGN**

In this module all the basic android User Interfaces that gets the data from user and provide form elements to input data, for welcome page,signup page, friend view location page etc.

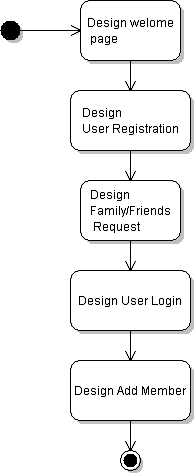


Fig 4.1 User Interface Design

**Module 2: GPS CONNECTIVITY : (SYNC UP OF LOCATION )**

In this module, the GPS a connected to application and GOOGLE API Key is obtained. Further the Coordinates are marked in map and updation of location is done often in such a way that location change is notified.

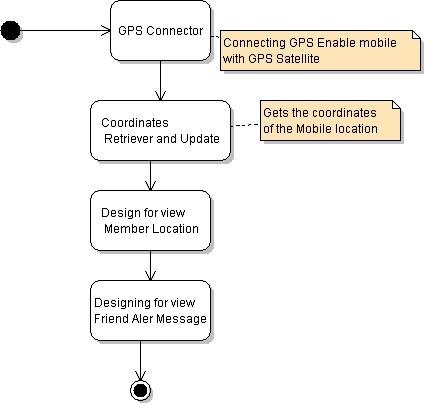


Fig 4.2 GPS Connectivity ( Sync up of Location )

**Module 3: SERVER APPLICATION**

This module take care of all the sever activities that includes handling the Registration, Login , Updating process , Alerting process , calculating distance etc.

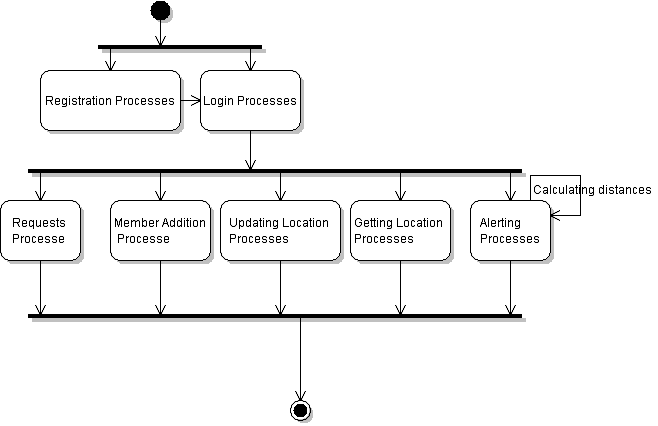


Fig 4.3 Server Application

**Module 4: DATABASE CONNECTIVITY FOR CLIENTS**

In this module a simple database is created using SQL Queries where the actual table stores the User Details and Geo Location Coordinates of Application user.Then the relationship between table entries are established. Finally the connectivity of database is made with the client.

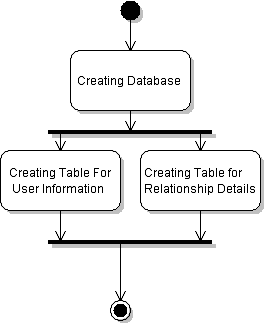


Fig 4.4 Database Connectivity

**CHAPTER 5**

**CONCLUSION AND FUTURE ENHANCEMENT**

**5.1 CONCLUSION**

The mobile location identifier will be very useful for user to check the continuous change in location. User can check the location of friend, family member or employer from anywhere with android and net connectivity.

The Security services and emergency services like 911 makes use of mobile network which sends the test signal to find the exact location. This application eliminates all the complexity and makes it as easier to get location. It will not affect the privacy of users because only the register users in a circle can see the location and not everyone else.

**5.2 FUTURE ENHANCEMENT**

Furthur Automatic Message sending can be initiated when the locations are nearby and contact retrieval can be made inorder to improve the simplicity. With the improvement in data connectivity, this project can be further enhanced in such a way that multiple persons location could be tracked and task can be assigned when one user is found to be in the location which is needed by other user through messaging and automatic notification can be assigned to the user when he/she comes to that particular location.

**APPENDIX-A**

**SAMPLE CODING**

**LAYOUTS**

Main.xml

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

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

android:orientation="vertical"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

>

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Welcome to Friend Finder"

android:textAppearance="?android:attr/textAppearanceLarge" />

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textAppearance="?android:attr/textAppearanceLarge" />

<Button android:text="Signup"

android:id="@+id/signup"

android:layout\_width="fill\_parent"

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

</Button>

<Button android:text="View Friend"

android:id="@+id/view1"

android:layout\_width="fill\_parent"

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

</Button>

</LinearLayout>

Signup.xml

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

<LinearLayout

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

android:orientation="vertical"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Static IP (ex.115.10.145.65)"

/>

<EditText

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:id="@+id/ipadd"

/>

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Mobile Number"

/>

<EditText

android:id="@+id/mobile"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:phoneNumber="true">

</EditText>

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="Password"

></TextView>

<EditText android:text=""

android:id="@+id/pass"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:password="true"

>

</EditText>

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="Latitude"

></TextView>

<EditText

android:id="@+id/latitude"

android:layout\_width="fill\_parent"

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

</EditText>

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="Longitude">

</TextView>

<EditText

android:id="@+id/longitude"

android:layout\_width="fill\_parent"

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

</EditText>

<Button android:text="Submit"

android:id="@+id/create"

android:layout\_width="fill\_parent"

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

</Button>

</LinearLayout>

Viewfriend.xml

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

<LinearLayout

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

android:orientation="vertical"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Static IP (ex.115.10.145.65)"

/>

<EditText

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:id="@+id/ipadd1"

/>

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="Enter Mobile Number"/>

<EditText

android:id="@+id/mobile123"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:phoneNumber="true"

/>

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="Enter Password"

/>

<EditText

android:id="@+id/pass123"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:password="true"

/>

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="Enter Your Friend's Phone Number"

/>

<EditText

android:id="@+id/friend123"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:phoneNumber="true"

/>

<Button

android:text="View"

android:id="@+id/viewfriend123"

android:layout\_width="fill\_parent"

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

</Button>

</LinearLayout>

Update.xml

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

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

android:orientation="vertical"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

>

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Static IP (ex.115.10.145.65)"

/>

<EditText

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:id="@+id/ipadd"

/>

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Mobile Number"

/>

<EditText

android:id="@+id/mobile"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:phoneNumber="true">

</EditText>

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Password"

/>

<EditText

android:id="@+id/password"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:password="true">

</EditText>

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Latitude"

/>

<EditText android:text=""

android:id="@+id/latitude"

android:layout\_width="fill\_parent"

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

</EditText>

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Longitude"

/>

<EditText android:text=""

android:id="@+id/longitude"

android:layout\_width="fill\_parent"

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

</EditText>

</LinearLayout>

**ANDROID MANIFEST**

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

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

package="com.friend"

android:versionCode="1"

android:versionName="1.0">

<application android:icon="@drawable/icon" android:label="@string/app\_name">

<activity android:name=".mainclass" android:label="@string/app\_name">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

<activity android:name=".friend" android:label="@string/app\_name"></activity>

<activity android:name=".Viewfriend" android:label="@string/app\_name"></activity>

</application>

<uses-permission android:name="android.permission.INTERNET"></uses-permission>

<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION"></uses-permission>

<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION"></uses-permission>

</manifest>

**ACTIVITIES :**

Mainclass.java

package com.friend;

public class mainclass extends Activity {

Button signup,view1;

@Override

public void onCreate(final Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

signup=(Button)findViewById(R.id.signup);

view1=(Button)findViewById(R.id.view1);

signup.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

Intent intent=new Intent(getBaseContext(),friend.class);

startActivity(intent);

}

});

view1.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

Intent intent=new Intent(getBaseContext(),Viewfriend.class);

startActivity(intent);

}

});

}

}

Signup.java

package com.friend;

import com.mysql.jdbc.Connection;

import com.mysql.jdbc.Statement;

public class friend extends Activity {

/\*\* Called when the activity is first created. \*/

EditText mobile,lon1,lat1,pass,ipadd;

Button create1;

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.signup);

mobile=(EditText)findViewById(R.id.mobile);

pass=(EditText)findViewById(R.id.pass);

lon1=(EditText)findViewById(R.id.longitude);

lat1=(EditText)findViewById(R.id.latitude);

ipadd=(EditText)findViewById(R.id.ipadd);

create1=(Button)findViewById(R.id.create);

LocationManager mlocManager = (LocationManager)getSystemService(Context.LOCATION\_SERVICE);

LocationListener mlocListener = new MyLocationListener();

mlocManager.requestLocationUpdates( LocationManager.GPS\_PROVIDER, 0, 0, mlocListener);

create1.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

try{

String mob=mobile.getText().toString();

String longi=lon1.getText().toString();

String lati=lat1.getText().toString();

String password=pass.getText().toString();

String ipaddr=ipadd.getText().toString();

String dburl="jdbc:mysql://"+ipaddr+":3306/friendmapper";

if((longi!=null)&&(lati!=null))

{

Class.forName("com.mysql.jdbc.Driver");

Connection con=DriverManager.getConnection(dburl,"root","password");

Statement ps1=con.createStatement();

java.util.Date dt=new java.util.Date();

String s=dt.~~toLocaleString~~();

int i=ps1.executeUpdate("insert into createacc(MobileNumber,Longitude,Lattitude,Time,Password) values('"+mob+"','"+longi+"','"+lati+"','"+s+"','"+password+"')");

if(i>0)

{

Toast.makeText(getApplicationContext(), "success" ,Toast.LENGTH\_LONG).show();

}

else

{

Toast.makeText(getApplicationContext(), "failed to install" ,Toast.LENGTH\_LONG).show();

}

}

}

catch(Exception ex1)

{

Toast.makeText(getApplicationContext(), ex1.getMessage() ,Toast.LENGTH\_LONG).show();

}

// try{

// Thread.sleep(50000);

// }

// catch(Exception e){}

}

}

);

}

public class MyLocationListener implements LocationListener

{

public void onLocationChanged(Location loc)

{

loc.getLatitude();

loc.getLongitude();

//String Text = "My current location is: " +"latitude = "+ loc.getLatitude() + "Longitude = " + loc.getLongitude();

Double lat=loc.getLatitude();

Double lon=loc.getLongitude();

lat1.setText(lat.toString());

lon1.setText(lon.toString());

//Toast.makeText( getApplicationContext(),Text,Toast.LENGTH\_SHORT).show();

}

public void onProviderDisabled(String provider)

{

Toast.makeText( getApplicationContext(),"Gps Disabled",Toast.LENGTH\_SHORT ).show();

}

public void onProviderEnabled(String provider)

{

Toast.makeText( getApplicationContext(), "Gps Enabled", Toast.LENGTH\_SHORT).show();

}

public void onStatusChanged(String provider, int status, Bundle extras)

{

}

}/\* End of Class MyLocationListener \*/

}

Viewfriend.java

package com.friend;

import com.mysql.jdbc.Connection;

import com.mysql.jdbc.ResultSet;

import com.mysql.jdbc.Statement;

public class Viewfriend extends Activity {

String mylat,mylong;

EditText mob123,pass123,friend123,ipadd1;

Button viewfriend123;

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.viewfriend);

mob123=(EditText)findViewById(R.id.mobile123);

friend123=(EditText)findViewById(R.id.friend123);

pass123=(EditText)findViewById(R.id.pass123);

ipadd1=(EditText)findViewById(R.id.ipadd1);

viewfriend123=(Button)findViewById(R.id.viewfriend123);

viewfriend123.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

LocationManager mlocManager = (LocationManager)getSystemService(Context.LOCATION\_SERVICE);

LocationListener mlocListener = new MyLocationListener();

mlocManager.requestLocationUpdates( LocationManager.GPS\_PROVIDER, 0, 0, mlocListener);

try {

String ipaddr=ipadd1.getText().toString();

String dburl="jdbc:mysql://"+ipaddr+":3306/friendmapper";

Class.forName("com.mysql.jdbc.Driver");

Connection con=DriverManager.getConnection(dburl,"root","password");

Statement ps1=con.createStatement();

ResultSet rs=ps1.executeQuery("select \* from createacc where MobileNumber='"+mob123.getText().toString()+"' and Password='"+pass123.getText().toString()+"'");

if(rs.next())

{

Statement ps2=con.createStatement();

ResultSet rs1=ps1.executeQuery("select \* from createacc where MobileNumber='"+friend123.getText().toString()+"'");

if(rs1.next())

{

Intent openLink = new Intent(Intent.ACTION\_VIEW, Uri.parse("https://maps.google.com/maps?q="+rs1.getString(3)+","+rs1.getString(2)+""));

startActivity(openLink);

}

}

else

{

Toast.makeText(getApplicationContext(), "mobile number or password is incorrect" ,Toast.LENGTH\_LONG).show();

}

} catch (Exception e) {

Toast.makeText(getApplicationContext(), e.getMessage() ,Toast.LENGTH\_LONG).show();

e.printStackTrace();

}

}

});

}

public class MyLocationListener implements LocationListener

{

public void onLocationChanged(Location loc)

{

loc.getLatitude();

loc.getLongitude();

//String Text = "My current location is: " +"latitude = "+ loc.getLatitude() + "Longitude = " + loc.getLongitude();

Double lat=loc.getLatitude();

Double lon=loc.getLongitude();

// lat1.setText(lat.toString());

// lon1.setText(lon.toString());

mylat=lat.toString();

mylong=lon.toString();

//Toast.makeText( getApplicationContext(),Text,Toast.LENGTH\_SHORT).show();

}

public void onProviderDisabled(String provider)

{

Toast.makeText( getApplicationContext(),"Gps Disabled",Toast.LENGTH\_SHORT ).show();

}

public void onProviderEnabled(String provider)

{

Toast.makeText( getApplicationContext(), "Gps Enabled", Toast.LENGTH\_SHORT).show();

}

public void onStatusChanged(String provider, int status, Bundle extras)

{

}

}/\* End of Class MyLocationListener \*/

}

Friendrun.java

package com.friend1;

public class friendrun extends Activity {

/\*\* Called when the activity is first created. \*/

EditText mobile,lon1,lat1,pw,ipadd;

TimerTask tim;

Timer t=new Timer();

final Handler hand=new Handler();

int x=1;

@Override

public void onCreate(final Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

tim=new TimerTask(){

public void run()

{

hand.post(new Runnable() {

public void run() {

mobile=(EditText)findViewById(R.id.mobile);

pw=(EditText)findViewById(R.id.password);

lon1=(EditText)findViewById(R.id.longitude);

lat1=(EditText)findViewById(R.id.latitude);

ipadd=(EditText)findViewById(R.id.ipadd);

LocationManager mlocManager = (LocationManager)getSystemService(Context.LOCATION\_SERVICE);

LocationListener mlocListener = new MyLocationListener();

mlocManager.requestLocationUpdates( LocationManager.GPS\_PROVIDER, 0, 0, mlocListener);

// create1.setOnClickListener(new OnClickListener() {

// @Override

// public void onClick(View v) {

try{

String mob=mobile.getText().toString();

String longi=lon1.getText().toString();

String lati=lat1.getText().toString();

String pw1=pw.getText().toString();

String ipaddr=ipadd.getText().toString();

String dburl="jdbc:mysql://"+ipaddr+":3306/friendmapper";

if((longi!=null)&&(lati!=null))

{

Class.forName("com.mysql.jdbc.Driver");

Connection con=DriverManager.getConnection(dburl,"root","password");

Statement ps1=con.createStatement();

java.util.Date dt=new java.util.Date();

String s=dt.~~toLocaleString~~();

int i=ps1.executeUpdate("update createacc set Longitude='"+longi+"',Lattitude='"+lati+"',Time='"+s+"' where MobileNumber='"+mob+"' and Password='"+pw1+"'");

if(i>0)

{

//Toast.makeText(getApplicationContext(), "success" ,Toast.LENGTH\_LONG).show();

}

else

{

Toast.makeText(getApplicationContext(), "your account does not exist. please create a account." ,Toast.LENGTH\_LONG).show();

}

}

else

{

Toast.makeText(getApplicationContext(), "waiting for gps values" ,Toast.LENGTH\_LONG).show();

}

}

catch(Exception ex1)

{

Toast.makeText(getApplicationContext(), "Account already exist" ,Toast.LENGTH\_LONG).show();

}

//

}});

}};

t.schedule(tim, 900, 60000);

}

public class MyLocationListener implements LocationListener

{

public void onLocationChanged(Location loc)

{

loc.getLatitude();

loc.getLongitude();

//String Text = "My current location is: " +"latitude = "+ loc.getLatitude() + "Longitude = " + loc.getLongitude();

Double lat=loc.getLatitude();

Double lon=loc.getLongitude();

lat1.setText(lat.toString());

lon1.setText(lon.toString());

//Toast.makeText( getApplicationContext(),Text,Toast.LENGTH\_SHORT).show();

}

public void onProviderDisabled(String provider)

{

Toast.makeText( getApplicationContext(),"Gps Disabled",Toast.LENGTH\_SHORT ).show();

}

public void onProviderEnabled(String provider)

{

Toast.makeText( getApplicationContext(), "Gps Enabled", Toast.LENGTH\_SHORT).show();

}

public void onStatusChanged(String provider, int status, Bundle extras)

{

}

}/\* End of Class MyLocationListener \*/

}

**APPENDIX-B**

**SNAPSHOTS**

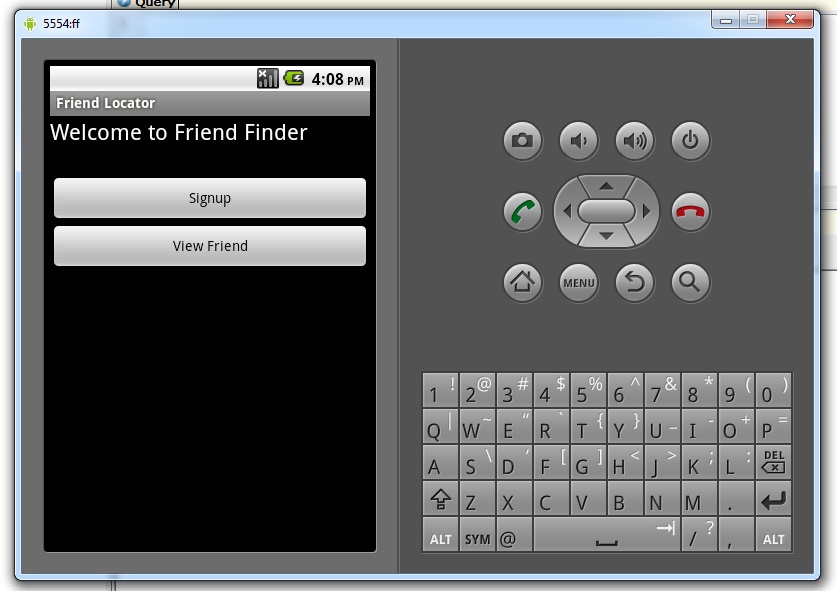
****

Fig B.1 **Welcome screen**

The above Snapshot is the welcome screen of the application. If the user is entering the first time, he/she has to sign up or straight away the friends can be viewed

****

Fig B.2 Sign Up Page

The above Snapshot allows the user to signup when the user enters the application for the first time, user can create a username and a password. The latitude and longitude positions are get automatically and once submit is pressed it is sent to database.

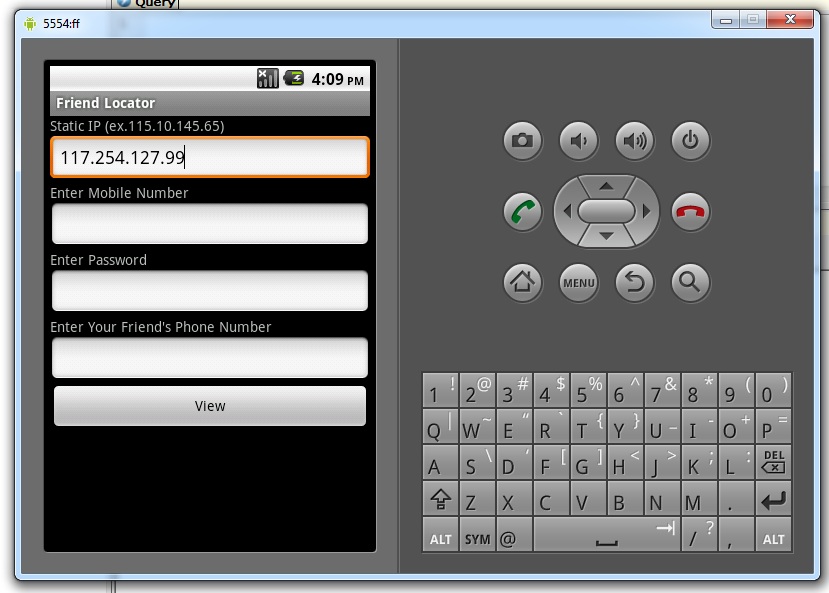
****

Fig B.3 **View friendPage**

The above snapshot allows to send request to server to view a users location. It is done by sending the other users mobile number along with the user name and password of finder.

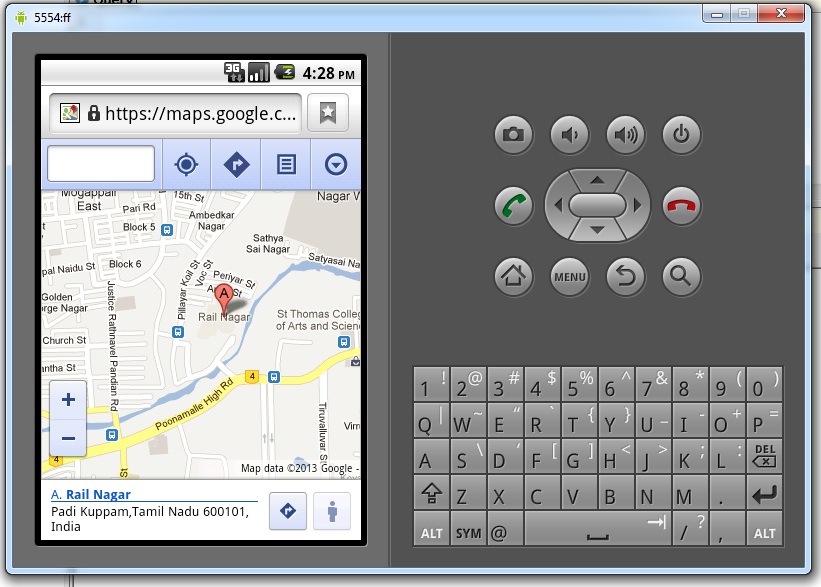


Fig B.4 Map View

This snapshot is the final output of the project. It pin points the location of other user in a Google Map.

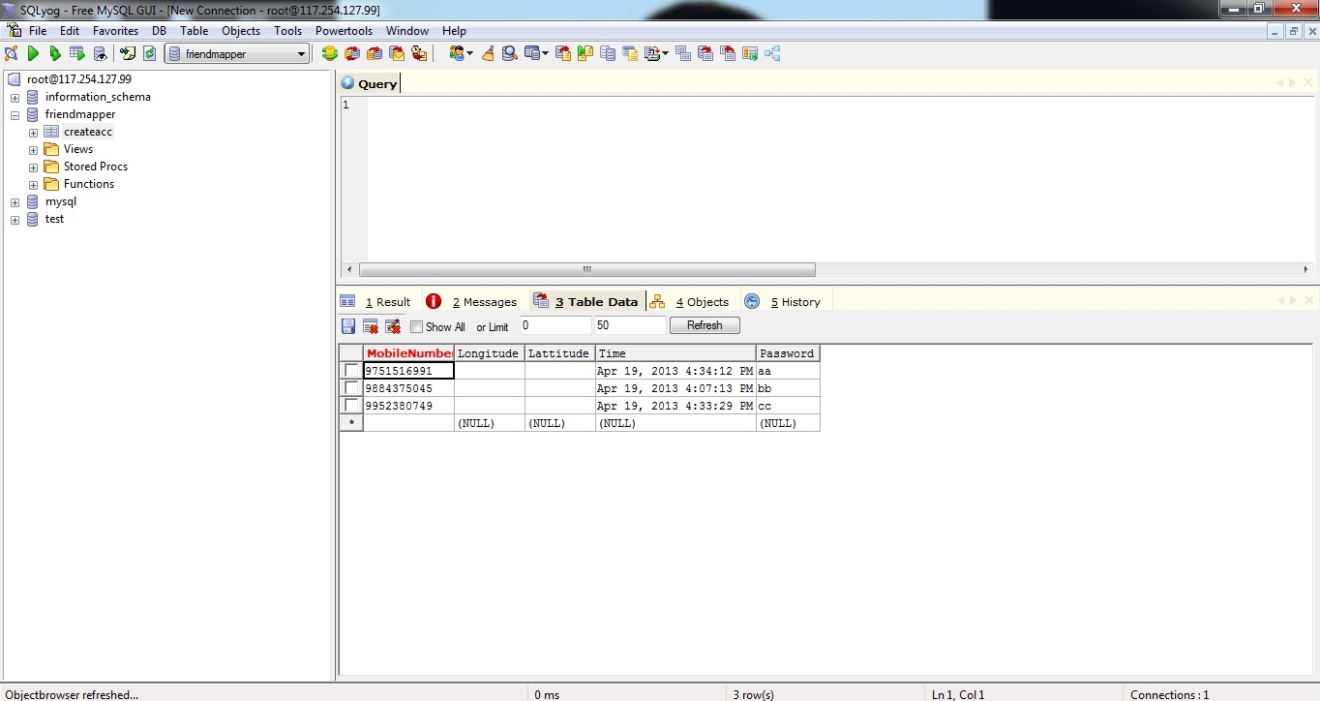


Fig B.5 Server Application

The above Snapshot shows the SQL server database in which the entries are added in table when a user signsup his information. This acts as a source for mobiles to retrieve information.

**REFERENCES**

[1]Locating Friends and Family Using Mobile GlobalPositioning System (GPS) Ghaith Bader Al-Suwaidi, Mohamed Jamal Zemerly

[2] LifeMap: A Smartphone-Based Context Provider for Location-Based

Services Yohan Chon and Hojung Cha Yonsei University

[3] A Location-Aware Framework for Intelligent Real-Time Mobile Applications Sean J. Barbeau, Rafael A. Perez, Miguel A. Labrador,Alfredo J. Perez, Philip L. Winters,and Nevine Labib Georggi University of South Florida

[4]The Digital Travel Diary System using the Network Service Platform Soichiro Ushio, Yuka Ito, Kazunori Okada, Tomoki Kitahara, Hidenori Tsuji

[5]<http://developer.android.com>

[6] <http://vogella.com>