# INTRODUCTION

* 1. **INTRODUCTION**

Our application is based on mobile CRM concept which will help the user.

* + 1. **PURPOSE**

Aim of this project deal with finding tourist attractions, optimal path finding for tourist attraction, suggestions for way of transportation, and if the tourist is opting for Rented Vehicle then calculation of the fare using optimal path distance calculation provided by Google Maps API. This project also helps the tourist to lodge a complaint against the Tourist Guide’s, Vehicle Drivers for Diverting the tourist and charging him unfair tariff & finding out emergency numbers for the particular city.

* + 1. **OVERVIEW**

Using this application user can detect source of user and from there he can get nearest tourist places in his area. User also can find the multiple routes, available transport facility and fair calculation. The data found on this app is more than Google. In current scenario some application shows the data which only available on Google, in these app there is no local places on Google, so we are trying to give the data of local places also which are not much popular on Internet with detail information.

* + 1. **BUSINESS CONTEXT**

1. Future Mobile Customer Relationship Management in the automotive industry and the tourism. The key profiles of future mobile communication are Interactive Broadband Protocols, Location Based Services and Individualized/Personalized Services mainly based on Multimedia information. These profiles are embedded in a three layer communicate model.
2. The grade of customer’s satisfaction is most relevant factor for the breakdown or the success of a company.
3. Aim of this project deal with finding tourist attractions, optimal path finding for tourist attraction, suggestions for way of transportation, and if the tourist is opting for Rented Vehicle then calculation of the fare using optimal path distance calculation provided by Google Maps API.
4. This project also helps the tourist to lodge a complaint against the Tourist Guide’s, Rented Vehicle Drivers for diverting the tourist and charging him unfair tariff & finding out emergency numbers for the particular city.
   1. **PROBLEM STATEMENT**

To develop an android application for tourists who are exploring the city. To help the tourists search for places and provide security like lodging complaint against the drivers. To detect the distance of the destination from the source using distance algorithm.

* 1. **PROJECT SCOPE**

This project will aim at developing an android application which will help the user to locate the nearby places. The application will use the Goople API to display the places and distance algorithm to calculate the distance from source to destination.

When the user opens the application source of the user will be detected. The user can select the categories which he wants to visit.This will show the places according to the categories selected. The places will also be shown as per the current weather. The algorithm will calculate the fare as per the distance.

According to weather places will be suggested to the user also.

* 1. **PROJECT OBJECTIVES**

Our project aims at delivering an application to the customer where he can use it whenever he visits the city. Project objectives are:

1. Detecting the source of the user.
2. Listing of tourist places.
3. Calculating fare according to the distance.
4. Lodge a compliant.
5. Show places according to the weather.
6. Places are fetched from database as well as Google.
7. Places can be sorted as per the categories.
   1. **ASSUMPTIONS AND DEPENDENCIES**

For the application to run following constraints should be present:-

* + 1. **Assumptions**

1. The device on which the application is running should be a android device.
2. User must have basic knowledge to operate android phone.
   * 1. **Dependencies**
3. The device should have the facility of GPS.
4. The device should be connected to the internet.
   1. **LITERATURE SURVEY**

Literature survey is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy n company strength. Once these things are satisfied, ten next steps are to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system the above consideration are taken into account for developing the proposed system.

* + 1. **Customer Relationship Management Using Android Phone in Tourism**

**Authors: Nitin Khondre, Ravi Saini, Ronak Jain, Sarang Suryawanshi, Bushra Quazi**

**Year: March 2014**

**Journal: International Journal of Emerging Technology and Advanced Engineering**

**Website: www.ijetae.com (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 4, Issue 3, March 2014)**

Customers are the vital key for each business and company to help them to grow. So, implementing CRM important tools that will help managers and companies to increase the satisfaction and loyalty of customers more than before. Nowadays it is very difficult for a company to convince a customer with only product or price arguments because of the strong competition in almost all market areas. Mobile technology offers a high potential to significantly transform the ways how a company can interact with their customers and even with own employees. Therefore, this paper deals with the possibilities and aspects to support CRM via future mobile services. [1]

* + 1. **inGuide-Interactive Guide**

**Authors: Filipe Andre Gomes Batista, Nuno Rodrigues, and Alexandrino Goncalves**

**Year: 2009**

**Journal: 3rd IEEE International Conference on Digital Ecosystems and Technologies Future Mobile CRM in Automotive and Tourist Area**

This paper describes the inGuide modular application which provides a package management system avoiding the need for a different version of the application for each city. It also describes the geolocation technology in order to provide contextual information in a simple and interactive way. This paper describes two modes those are online mode and offline mode. We preferred online mode of GPS tracking as it gives more accurate location. [2]

* + 1. **On-line GPS Track Simplification Algorithm for Mobile Platforms**

**Author: R. Ivanov**

**Year: 2010**

**Journal: Information Technology and Control**

This paper describes an algorithm for on-line simplification of the number of points, describing a GPS track. It is offered on the base of analysis of the location of three last points and calculation by basic trigonometric ratios and distance formula. [3]

* + 1. **Overview on Android- The New Mobile Operating System**

**Author: Monika Bazard, Sonia Bhardwaj**

**Year: April, 2011**

**Journal: SGI Reflections- International Journal of Science, Technology and Management. ISSN No. 0976-2140. Volume 2, Issue 1, April, 2011**

This paper describes the Android’s history, architecture, libraries and its advantages and disadvantages in the smart phones. [4]

**2. PROJECT PLAN**

**2.1 TASK SHEET SCHEDULE**

|  |  |  |  |
| --- | --- | --- | --- |
| **TASK NAME** | **TASK DURATION** | **START DATE** | **END DATE** |
| Search for BE project topics and related papers |  |  |  |
| Short listing of topics |  |  |  |
| Presentation of topic to the project coordinator and final selection of topic |  |  |  |
| Submission of base paper and synopsis | 2 days | 30/08/2014 | 1/09/2014 |
| Discussion on SRS and implementation of SRS | 2 days | 9/09/2014 | 11/09/2014 |
| Corrections in SRS and implementation of UML diagram | 2 days | 18/09/2014 | 20/09/2014 |
| Discussion on literature survey and SRS format | 2 days |  | 25/09/2014 |
| Preparation of Partial report in latex | 8 days |  | 9/10/2014 |
| Partial report submission and signing | 1 day | 17/10/2014 | 17/10/2014 |
| Semester VII project viva | 1 day | /10/2014 | /10/2014 |
| Discussion on what to do in semester VIII | 1 day | 22/12/2014 | 22/12/2014 |
| Changes in Admin Module and rough ER | 2 days | 09/01/2015 | 11/01/2015 |
| Part of Android application discussed with project guide | 5 days | 14/01/2015 | 19/01/2015 |
| Working on database | 15 days | 08/01/2015 | 23/01/2015 |
| Completion of Admin module | 15 days | 08/01/2015 | 23/01/2015 |
| Review I of project | 1 day | 28/01/2015 | 28/01/2015 |
| Discussion on class diagram | 1 day | 23/02/2015 | 23/02/2015 |
| Preparation of Final report in latex | 11 days | 1/03/2015 | 11/03/2015 |
| Shown Android module to the project guide | 1 day | 16/03/2015 | 16/03/2015 |
| Showed the changes in the project to the project guide | 8 days | 20/03/2015 | 27/03/2015 |
| Review II of project | 1 day | 28/03/2015 | 28/03/2015 |
| Discussion on some points of report and changes in the report | 2 days | 28/03/2015 | 30/03/2015 |
| Final Report submission signing | 1 day | 15/04/2015 | 15/04/2015 |

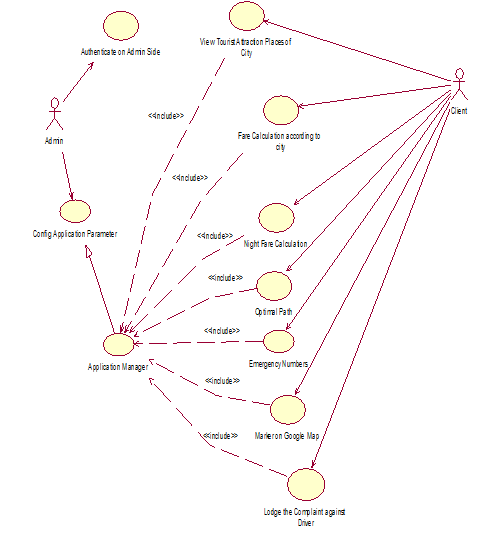
* 1. **Task Sheet Schedule**

1. **REQUIREMENT ANALYSIS**

**3.1 HARDWARE REQUIREMENTS**

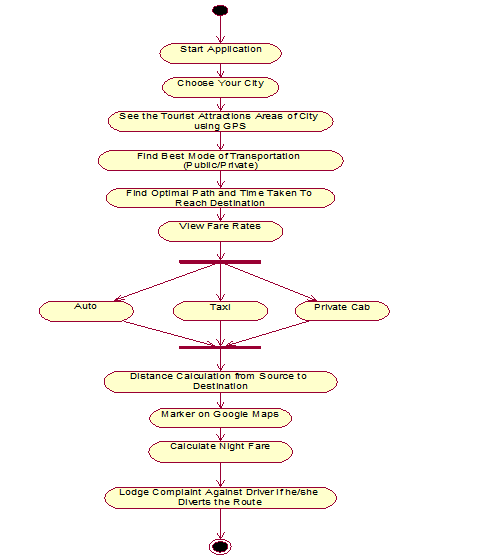
The system requires following hardware requirements:

1. System: Intel P4, 2.4 GHZ, 40 GB HDD for installation.
2. Memory: 512 MB memory, 256 MB ram
3. Project’s server side system is windows based supporting versions windows XP onwards.
   1. **SOFTWARE REQUIREMENTS**
4. Eclipse 3.7 Indigo
5. Android SDK
6. Android 2.3
7. Android GPS API
8. Apache Tomcat Server
9. MySQL
10. **PROJECT DESIGN**
    1. **E-R DIAGRAM**
    2. **DFD DIAGRAM**
       1. **DFD Level 0**
       2. **DFD Level 1**
    3. **UML DIAGRAMS**
       1. **Usecase Diagram**

****

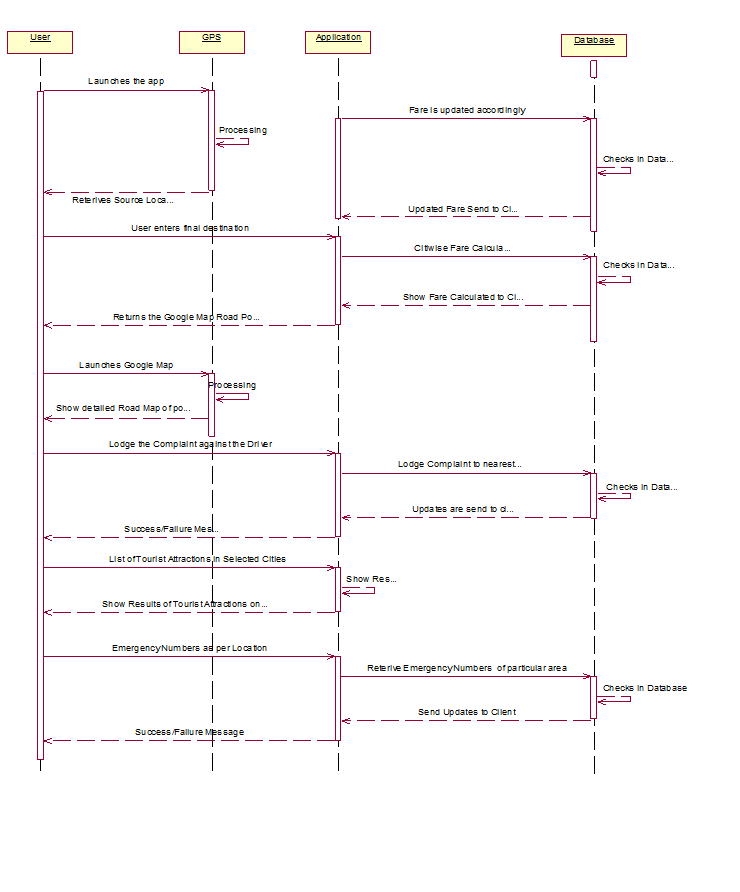
**4.3.1 Usecase Diagram**

* + 1. **Class Diagram**
    2. **Activity Diagram**

****

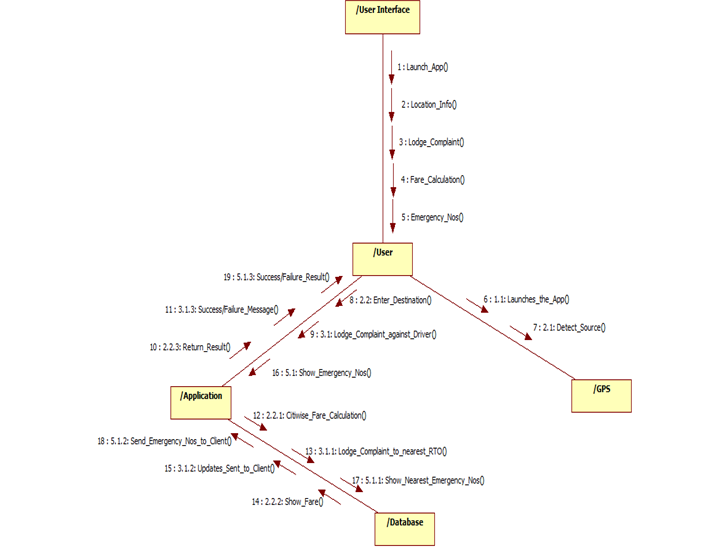
**4.3.3 Activity Diagram**

* + 1. **Package Diagram**
    2. **Sequence Diagram**

****

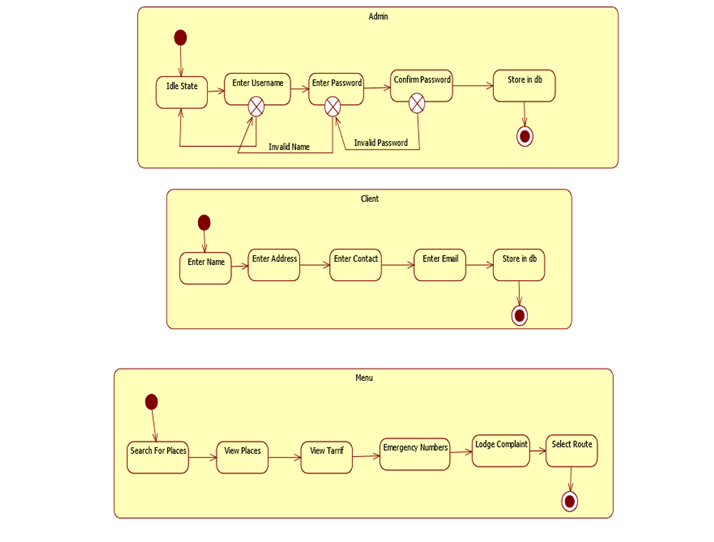
**4.3.5 Sequence Diagram**

* + 1. **Communication Diagram**

****

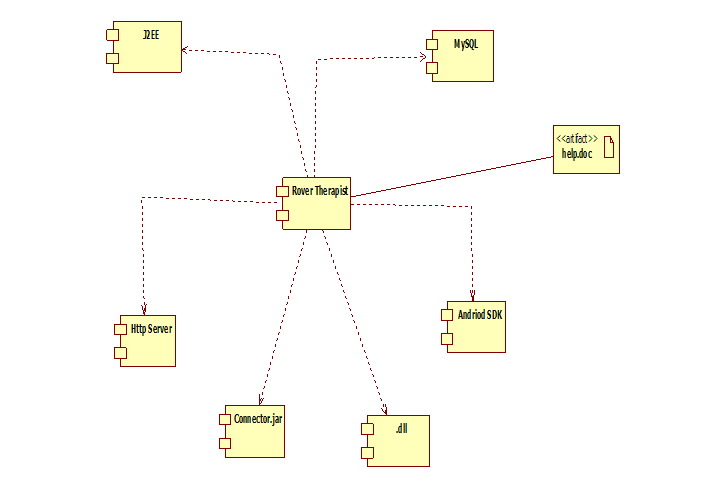
**4.3.6 Communication Diagram**

* + 1. **Composite Structure Diagram**
    2. **State Machine Diagram**

****

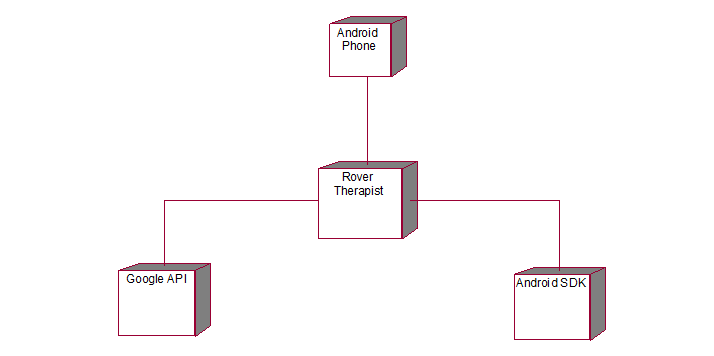
**4.3.8 State Machine Diagram**

* + 1. **Component Diagram**

****

**4.3.9 Component Diagram**

* + 1. **Deployment Diagram**

****

**4.3.10 Deployment Diagram**

1. **IMPLEMENTATION DETAILS**
   1. **PROJECT ARCHITECTURE**

Our application has a three-tier architecture. It has three layers:

1. Presentation Layer

It provides user interface. It handles the interaction with the user.

1. Logic Layer

Contains the business logic.

1. Data Layer

It is the physical storage layer for data persistence. It manages access to the database.

* 1. **ALGORITHM**

**DISTANCE BASED ALGORITHM**

The distance based algorithm used is the Hoversine algorithm. Hoversine algorithm uses the latitude and longitude for calculation of distance from the GPS. It calculates the shortest distance between the two points.

Haversine Formula:

a=(+cos.cos.(

c=2.atan2( )

d=R.c

where is latitude,

is longitude,

R is earth’s radius (mean radius =6,371 km)

Implementation of Algorithm:

**public** **static** **double** calcDistance(**double** lat\_a, **double** lng\_a, **double** lat\_b,**double** lng\_b) {

**float** pk = (**float**) (180 / Math.*PI*);

**double** a1 = lat\_a / pk;

**double** a2 = lng\_a / pk;

**double** b1 = lat\_b / pk;

**double** b2 = lng\_b / pk;

**double** t1 = (**double**) (Math.*cos*(a1) \* Math.*cos*(a2) \* Math.*cos*(b1) \* Math.*cos*(b2));

**double** t2 = (**double**) (Math.*cos*(a1) \* Math.*sin*(a2) \* Math.*cos*(b1) \* Math.*sin*(b2));

**double** t3 = (**double**) (Math.*sin*(a1) \* Math.*sin*(b1));

**double** tt = Math.*acos*(t1 + t2 + t3);

**return** 6366000 \* tt;

}

* 1. **TECHNOLOGIES, TOOLS AND LIBRARIES USED**
     1. **TECHNOLOGIES**
        1. **Java**

Java is a general-purpose computer language that is concurrent, class-based, object-oriented. It contains features like classes, objects, encapsulation, abstraction, inheritance and polymorphism. Java is designed by James Gosling and Sun Microsystems. Java is simple, robust, secure, system independent language, portability, interpreted, multithreaded. Sun Microsystems Inc. has divided Java into three parts - Java SE, Java EE and Java ME.

1. **Java SE:**

It is the Java Standard Edition that contains basic core java classes. This edition is used to develop standard applets and applications.

1. **Java EE:**

It is the Java Enterprise Edition and it contains classes that are beyond Java SE. To use many of the classes in Java EE, Java SE is used. It mainly concentrates on providing business solutions on a network.

1. **Java ME**

It stands for Java Micro Edition. It is for developers who develop code for portable devices, such as aPDA or a cellular phone.

* + - 1. **HTML**

HTML is a markup language commonly used to create Web pages. A markup language provides a way to describe the structure of text and graphics on a Web page. It is developed and maintained by World Wide Web consortium (W3C). The term hyper signifies the navigation from one location to another in a non-linear fashion. HTML defines the content, i.e. the structure and the layout of a Web page with the help of elements and attributes. An element includes the start and the end tags, with some content within them, and attributes provide additional information about the elements.

* + - 1. **CSS**

CSS is a style sheet language that is used to describe the apperance and formatting of a Web document, which is written in a markup language. It enables one to keep separate the instructions related to the presentation of Web content fron the Web content itself.

A CSS style sheet consists of a list of rules, which in turn consists of one or more selectors and a declaration block. Selectors are used to declare the marku elements to which a style applies to, while a declaration block consists of a list of declarations in braces.

* + - 1. **Servlet**

A servlet is a simple Java class, which is dynamically loaded on a Web server and thereby enhances the functionality of the Web server. Servlets are secure and portable as they run on JVM embedded with the Web server and cannot operate outside the domain of the Web server. That is servlets are objects that generate dynamic content after processing requests that originate from a Web browser. They are Java components that are used to create dynamic Web applications. Servlets can run on any Java-enabled platform and are usually designed to process HTTP requests, such as GET and POST.

* + - 1. **Javascript**

JavaScript is an object-oriented scripting language that is used to design interactive websites. It is developed by Netscape and works in all major browsers, such as Internet Explorer, Firefox, Chrome, Opera, and Safari. It is used with HTML code to add dynamic content to their websites. It is an interpreted language.

* + - 1. **Android**

Android is an operating system based on Linux with Java programming interface. It provides tools such as a compiler, debugger and a device emulator as well as JVM. It is created by the Open Handset Alliance which is lead by Google.

Android uses a special virtual machine, e.g. the Dalvik Virtual Machine. dalvik uses special bytecode. Therefore one cannot run standard of Java bytecode on Android. It provides a tool "dx" which allows conerting Java Class files into "dex" files. Android applications are packed into an .apk(Android Package) file.

Every Android application runs in its own process and under its own user id which is generated automatically by the Android system during deployment. Therefore the application is isolated from other running applications and a misbehaving applications cannot easily harm other Android application.

**ANDROID DEVELOPMENT TOOLS**

Google provides the ADT to develope Android applications with Eclipse. ADT is a set of components (plug-ins) which extend the Eclipse IDE with Android development capabilities.

ADT contains all required functionalities to create, compile, debug and deploy Android applications from the Eclipde IDE. It also allows creating and starting AVDs.

**ANDROID SDK**

The Android Software Development Kit(SDK) contains the necessary tools to create, compile and package Android application. Most of these tools are command line based.

It also provides an Android device emulator, so that Android application can be tested without a real Android phone. AVD (Android Virtual device) via the Android SDK can be created, which run in the emulator.

It contains the Android debug bridge (adb) tool which allows to connect to an virtual or real Android device.

* + - 1. **JSON(JavaScript Object Notation)**

JSON is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

JSON is built on two structures:

* A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.
* An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.

Example of JSON describing a person:

{

“firstName” : “John” ,

“lastName” : “Smith” ,

“isAlive” : true ,

“age” : 25 ,

“height\_cm” : 167.6 ,

“address” : {

“streetAddress” : “21 2nd Street” ,

“city” : “New York” ,

“state” : “NY” ,

“postalCode” : “10021-3100”

} ,

“phoneNumbers” : [

{

“type” : ”home” ,

“number” :”212 555-4567”

}

],

“children” : [ ] ,

“spouse” : null

}

* + 1. **TOOLS**
       1. **Eclipse**

Eclipse is an IDE. It contains a base workspace and an extensible plug-in system for customizing the environment. Written mostly in Java, it can be used to develop applications.

* + - 1. **MySQL**

MySQL is the world's second most widely used relational database management system (RDBMS) and most widely used open-source RDBMS. MySQL is a key part of LAMP (Linux, Apache, MySQL, PHP/ Perl/ Python), the fast-growing open source enterprise software stack. It uses SQL which is the most popular language for adding, accessing and managing content in a database.

* + - 1. **Apache TomCat Server**

Apache Tomcat is an open-source web server and servlet. Tomcat implements several Java Servlet, JavaServer Pages(JSP), Java EL, and WebSocket, and provides a "pure Java" HTTP web server environment for Java code to run in.

* + 1. **LIBRARIES**

1. mysql-connector-java-3.1.14-bin.jar
2. google-play-services.jar
3. json-jena-1.0.jar
4. android-support-v4.jar
5. gcm.jar
6. gson.jar
7. common-dbcp-1.4.jar
8. common-dbutilis-1.4.jar
   1. **DATABASE DETAILS**
9. **Domain Table**

CREATE TABLE `domain` (`domainId` int(10) unsigned NOT NULL auto\_increment, `domainDesc` varchar(45) NOT NULL, PRIMARY KEY (`domainId`));

1. **Domaininfo Table**

CREATE TABLE `domaininfo` (`iddomainInfo` int(10) unsigned NOT NULL auto\_increment, `domainId` int(10) unsigned NOT NULL, `description` longtext NOT NULL, `location` varchar(255) NOT NULL, `updatedDate` timestamp NOT NULL default CURRENT\_TIMESTAMP, `photourl` varchar(255) default NULL, `pincode` varchar(10) default NULL, `latitude` varchar(100) NOT NULL, `longitude` varchar(100) NOT NULL,`rating` int(11) NOT NULL, `ratinguserName` varchar(1000) NOT NULL, `phoneNumber` varchar(15) NOT NULL, `review` varchar(4000) NOT NULL, PRIMARY KEY (`iddomainInfo`));

1. **Smsmanger Table**

CREATE TABLE `smsmaneger` (`id` int(10) unsigned NOT NULL auto\_increment, `phoneno` varchar(45) NOT NULL, `msg` varchar(45) NOT NULL, PRIMARY KEY (`id`));

1. **Trackeruser Table**

CREATE TABLE `trackuser` ( `userid` int(10) unsigned NOT NULL, `lat` varchar(45) NOT NULL, `longitude` varchar(45) NOT NULL, `cellid` varchar(45) NOT NULL, `lac` varchar(45) NOT NULL, `udate` timestamp NOT NULL default CURRENT\_TIMESTAMP);

1. **Useraccount Table**

CREATE TABLE `useraccount` (`userid` int(10) unsigned NOT NULL auto\_increment, `password` varchar(45) default NULL, `imei` varchar(45) default NULL, `ipaddress` varchar(45) default NULL, `domainId` int(10) unsigned default '1', `displayName` varchar(45) default NULL, `cellId` varchar(45) default NULL, `lac` varchar(45) default NULL, `lat` varchar(45) default NULL, `longitude` varchar(45) default NULL, `activeFlag` varchar(45) default 'Y', `udate` timestamp NOT NULL default CURRENT\_TIMESTAMP, `mailid` varchar(100) default NULL, `phoneno` varchar(45) default NULL, `photourl` varchar(255) default NULL, `fathercontacts` varchar(255) default NULL, `username` varchar(45) default NULL, `adminflag` varchar(45) default 'N', PRIMARY KEY (`userid`));

1. **Usergeotags Table**

CREATE TABLE `usergeotags` (`geotagid` int(10) unsigned NOT NULL auto\_increment, `geoTagName` varchar(45) NOT NULL, `preference` varchar(45) NOT NULL, `actiondesc` varchar(45) NOT NULL, `lat` varchar(45) NOT NULL, `lng` varchar(45) NOT NULL, `imei` varchar(45) NOT NULL, PRIMARY KEY (`geotagid`));

* 1. **INTERFACE DETAILS**

Application will have the following interfaces:

* User interface screen will be choosing the attractions.
* User interface screen for choosing the distance.
* User interface screen for giving the input of the taxi fare.
* User interface screen for viewing place details.
* User interface screen for complaint launch.
* User interface screen for registering first time.
* User interface screen for showing location on the map.
* User interface screen for showing direction on the map with source and destination.
* User interface screen for showing different routes.
* User interface will provide good look and feel effect so that it will user friendly.
* And he or she can operate system very efficiently.
  1. **Screen Shots and Code**
     1. **Splash Screen**

****

* + - 1. **Splash Screen**

<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:background="@drawable/back"

tools:context=".Hint1" >

<TextView

android:id="@+id/textViewKM"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentLeft="true"

android:layout\_alignParentRight="true"

android:layout\_below="@+id/serverImageView"

android:gravity="center\_horizontal"

android:text="Future CRM"

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

android:textColor="#B1B1FF"

android:textSize="35dp"

android:textStyle="bold|italic"

android:typeface="serif" />

<ImageView

android:id="@+id/serverImageView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentLeft="true"

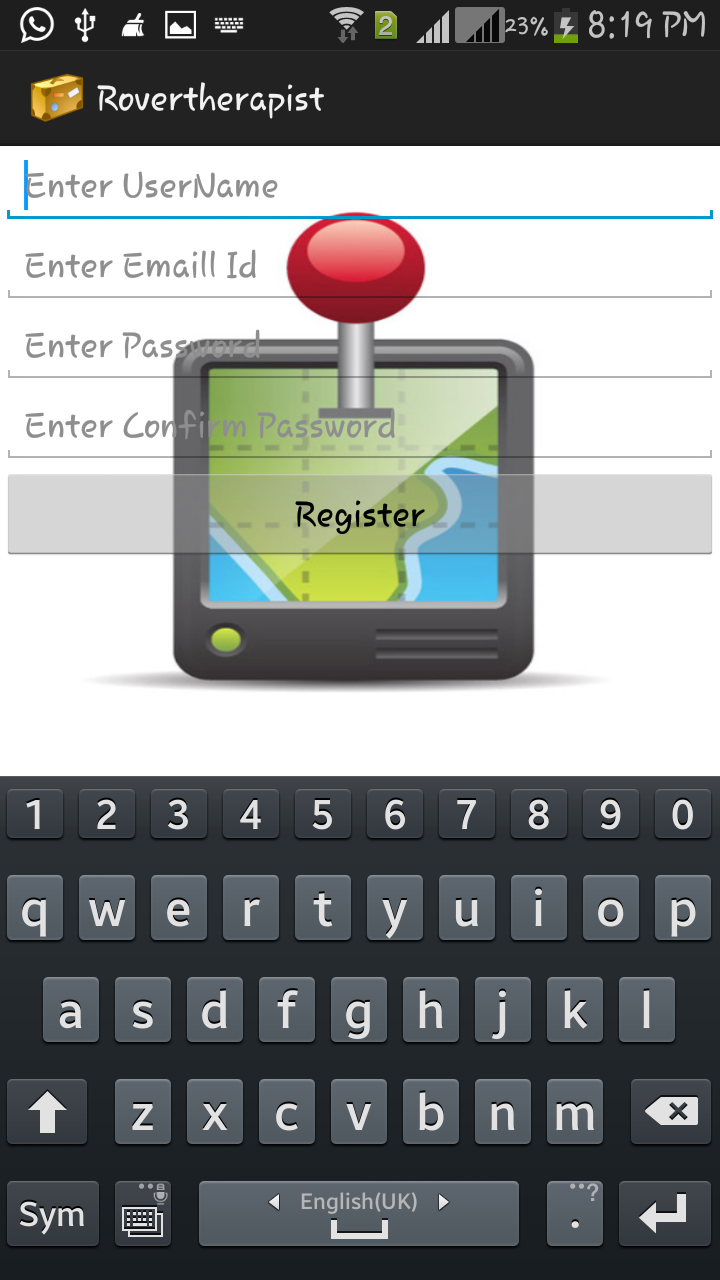
android:layout\_alignParentRight="true"

android:layout\_centerVertical="true"

android:src="@drawable/knapsack" />

</RelativeLayout>

* + 1. **Register User Screen**

****

**5.6.2 Register User Screen**

public class RegisterActivity extends CommonActivity {

EditText et, et1, et2, et3, et4;

AutoCompleteTextView autoCompleteLocation;

// Asyntask

AsyncTask<Void, Void, Void> mRegisterTask;

@Override

protected void onCreate(Bundle savedInstanceState) {

// TODO Auto-generated method stub

super.onCreate(savedInstanceState);

setContentView(R.layout.*register*);

et = (EditText) findViewById(R.id.*editText*);

et1 = (EditText) findViewById(R.id.*editText1*);

et3 = (EditText) findViewById(R.id.*editText3*);

et4 = (EditText) findViewById(R.id.*editText4*);

}

public void register(View v) {

String url = AndroidConstants.*MAIN\_URL*()

+ "method=insertUserDetailsObject";

String query = "displayname=" + et.getText() + "&useremail=" + et1.getText() + "&userpass=" + et3.getText() + "&imei=" + getIMEI();

url += "&" + query;

AndroidConstants.*EMAIL\_ID* = et1.getText().toString();

UserModel result = (UserModel) HttpView.*connect2ServerObject*(url);

if (result.isSuccess()) {

System.*out*.println("Calling Register User "+ et.getText().toString() + " " +et1.getText().toString());

go(WelcomeActivity.class);

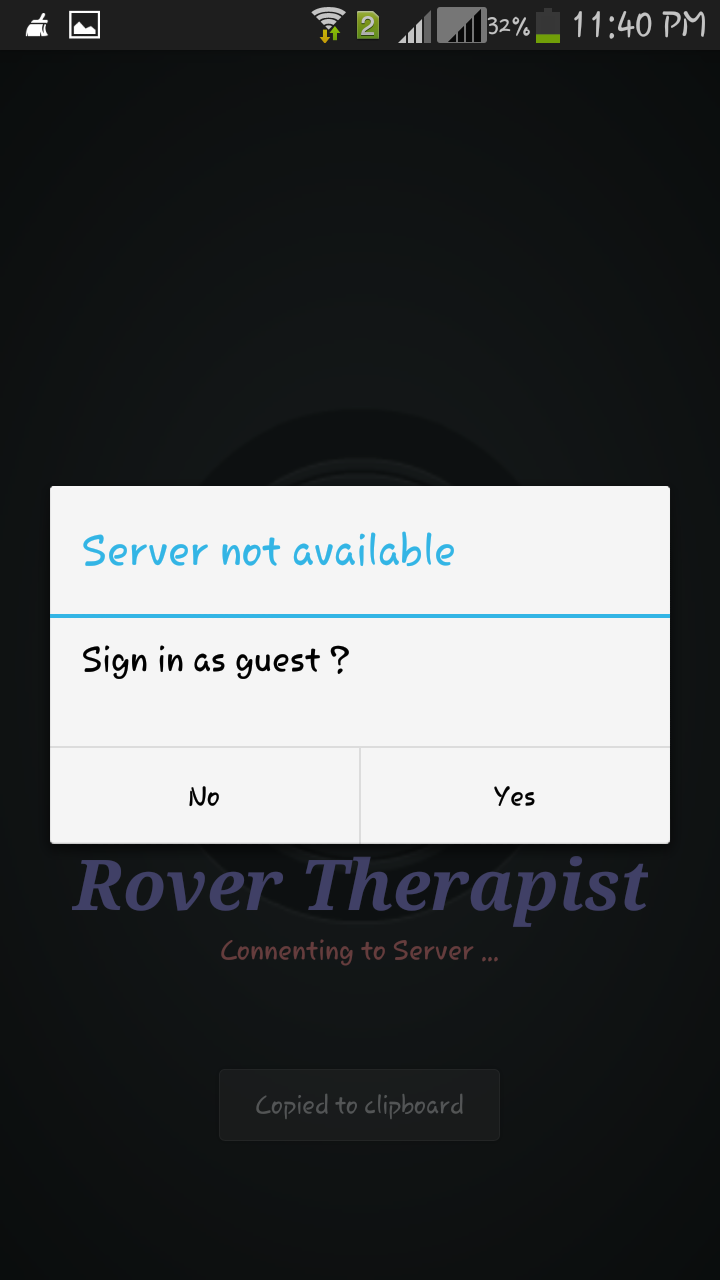
} else {

toast("User already exists");

}

}

* + 1. **Sign In As Guest Screen**

****

**5.6.3 Sign In As Guest Screen**

public void checkConnectivity() {

runOnUiThread(new Runnable() {

@Override

public void run() {

// TODO Auto-generated method stub

SharedPreferences s = PreferenceManager.getDefaultSharedPreferences(LoginActivity.this);

boolean success = checkConnectivityServer();

if (success) {

SharedPreferences.Editor editor = s.edit();

editor.putString("MAIN\_SERVER\_IP",AndroidConstants.MAIN\_SERVER\_IP + "");

editor.putString("MAIN\_SERVER\_PORT",AndroidConstants.MAIN\_SERVER\_PORT + "");

editor.commit();

checkUserDetails();

} else {

String MAIN\_SERVER\_IP=s.getString("MAIN\_SERVER\_IP",AndroidConstants.MAIN\_SERVER\_IP);

String MAIN\_SERVER\_PORT=s.getString("MAIN\_SERVER\_PORT",AndroidConstants.MAIN\_SERVER\_PORT);

if(!MAIN\_SERVER\_IP.equalsIgnoreCase(AndroidConstants.MAIN\_SERVER\_IP)|| !MAIN\_SERVER\_PORT.equalsIgnoreCase(AndroidConstants.MAIN\_SERVER\_PORT)) {

success = checkConnectivityServer(MAIN\_SERVER\_IP,

StringHelper.n2i(MAIN\_SERVER\_PORT));

if (success) {

AndroidConstants.MAIN\_SERVER\_IP = MAIN\_SERVER\_IP;

AndroidConstants.MAIN\_SERVER\_PORT = MAIN\_SERVER\_PORT;

checkUserDetails();

} else {

System.out.println("Redirecting to Config 1");

mHandler.post(new Runnable() {

@Override

public void run() {

AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(LoginActivity.this);

// set title

alertDialogBuilder.setTitle("Server not available ");

// set dialog message

alertDialogBuilder.setMessage("Sign in as guest ?").setCancelable(false).setPositiveButton(

"Yes",new DialogInterface.OnClickListener() {

public void onClick(

DialogInterface dialog,int id) {

go(RegisterActivity.class;

}

})

.setNegativeButton("No",new DialogInterface.OnClickListener() {

public void onClick(

DialogInterface dialog,int id) {

go(ConfigTabActivity.class);

}

});

// create alert dialog

AlertDialog alertDialog = alertDialogBuilder.create();

alertDialog.show();

}

});

// go(ConfigTabActivity.class);

}

} else {

System.out.println("Redirecting to Config 2");

mHandler.post(new Runnable() {

@Override

public void run() {

AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(LoginActivity.this);

// set title

alertDialogBuilder

.setTitle("Server not available ");

// set dialog message

alertDialogBuilder.setMessage("Sign in as guest ?").setCancelable(false).setPositiveButton(

"Yes",new DialogInterface.OnClickListener() {

public void onClick(

DialogInterface dialog,int id) {

go(HomeActivity.class);

}

})

.setNegativeButton("No",new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog,int id) {

go(ConfigTabActivity.class);

}

});

// create alert dialog

AlertDialog alertDialog = alertDialogBuilder.create();

alertDialog.show();

}

});

}

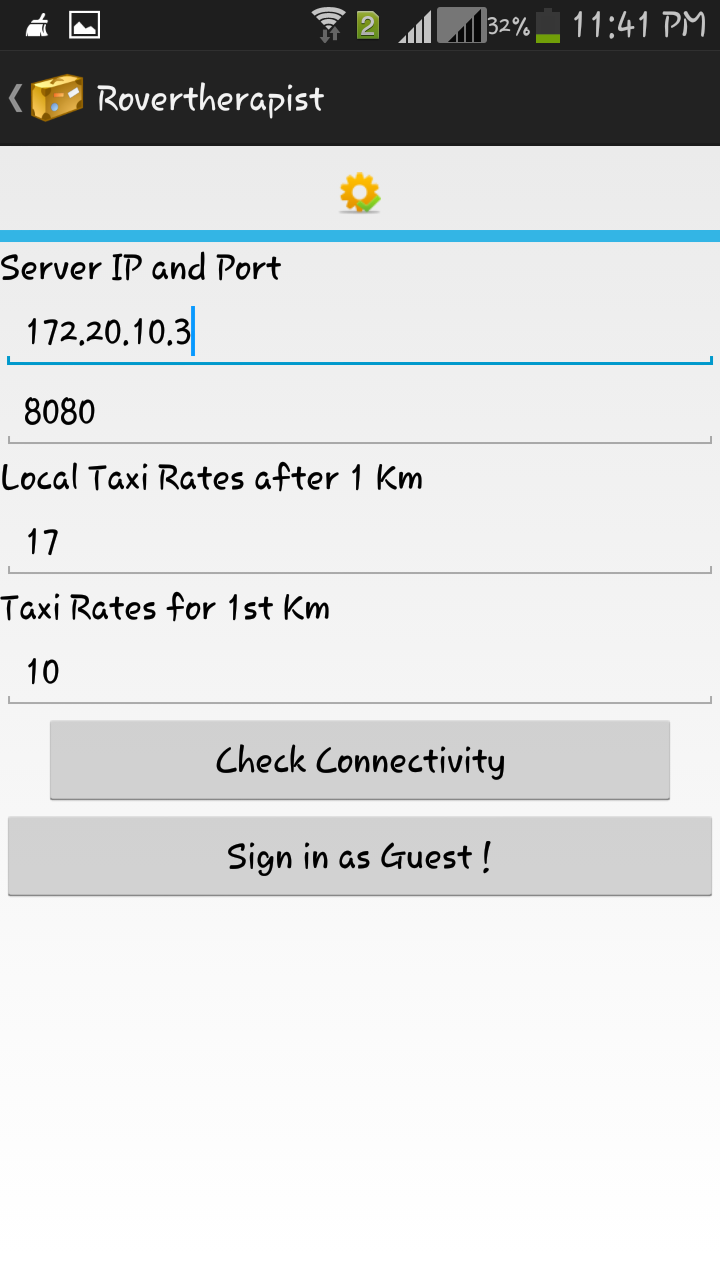
}

}

});

}

* + 1. **Server Connection Parameter Screen**

****

**5.6.4 Server Connection Parameter Screen**

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.config);

TabHost tabs = (TabHost) findViewById(R.id.tabHost);

android.os.StrictMode.ThreadPolicy tp = android.os.StrictMode.ThreadPolicy.LAX;

android.os.StrictMode.setThreadPolicy(tp);

tabs.setup();

TabHost.TabSpec spec1 = tabs.newTabSpec("tag1");

spec1.setContent(R.id.tab1);

spec1.setIndicator("", getResources().getDrawable(R.drawable.settings));

tabs.addTab(spec1);

// Server Settings

ipaddress = (EditText) findViewById(R.id.editText1);

port = (EditText) findViewById(R.id.editText2);

taxi1km = (EditText) findViewById(R.id.editText3);

taxirates = (EditText) findViewById(R.id.editText4);

SharedPreferences s = PreferenceManager

.getDefaultSharedPreferences(ConfigTabActivity.this);

ipaddress.setText(s.getString("MAIN\_SERVER\_IP",AndroidConstants.MAIN\_SERVER\_IP + ""));

port.setText(s.getString("MAIN\_SERVER\_PORT",AndroidConstants.MAIN\_SERVER\_PORT + ""));

taxi1km.setText(s.getString("taxiRates1stkm", "17"));

taxirates.setText(s.getString("taxirates", "10"));

}

public void toast(String message) {

Toast t = Toast.makeText(ConfigTabActivity.this, message,

Toast.LENGTH\_LONG);

t.show();

}

ProgressDialog progressDialog = null;

public void fnConfig(View v) {

String newIp = ipaddress.getText().toString().trim();

int newPort = StringHelper.nullObjectToIntegerEmpty(port.getText().toString());

SharedPreferences s = PreferenceManager

.getDefaultSharedPreferences(ConfigTabActivity.this);

SharedPreferences.Editor editor = s.edit();

editor.putString("MAIN\_SERVER\_IP", newIp);

editor.putString("MAIN\_SERVER\_PORT", String.valueOf(newPort));

editor.putString("taxiRates1stkm",String.valueOf(taxi1km.getText().toString()));

editor.putString("taxirates",String.valueOf(taxirates.getText().toString()));

editor.commit();

boolean success = LoginActivity.checkConnectivityServer(s.getString(

"MAIN\_SERVER\_IP", AndroidConstants.MAIN\_SERVER\_IP), Integer

.parseInt(s.getString("MAIN\_SERVER\_PORT",

AndroidConstants.MAIN\_SERVER\_PORT)));

if (success) {

go(LoginActivity.class);

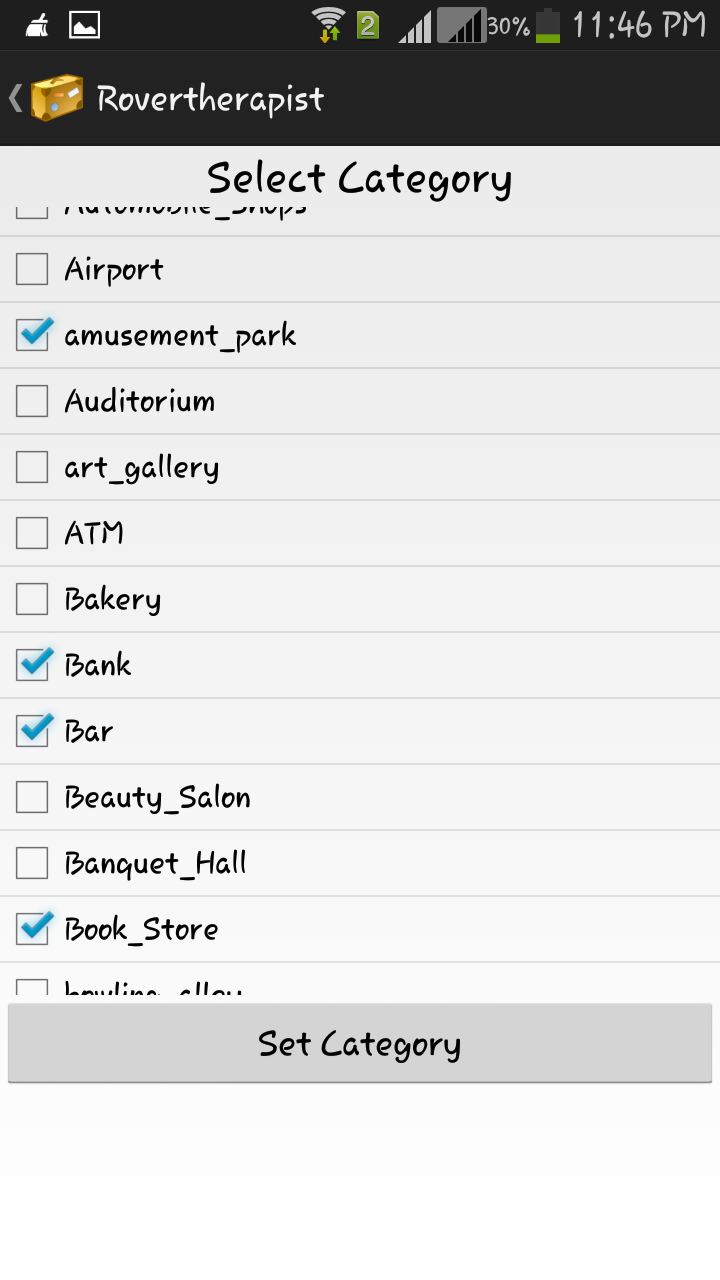
} else {

toast("Not able to connect");

}

}

* + 1. **Attraction List Screen**

****

**5.6.5 Attraction List Screen**

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_types\_list);

address = this.getIntent().getStringExtra("addr");

SharedPreferences s = PreferenceManager.getDefaultSharedPreferences(TypesList.this);

type = s.getString("types", CategoryConstants.ATTRACTIONS);

toast(type);

displayListView();

}

listView.setOnItemClickListener(new OnItemClickListener() {

public void onItemClick(AdapterView parent, View view,int position, long id) {

// When clicked, show a toast with the TextView text

TypeEntity country = (TypeEntity) parent.getItemAtPosition(position);

Toast.makeText(getApplicationContext(),"Clicked on Row: " + country.getName(),

Toast.LENGTH\_LONG).show();

}

* + 1. **Direction Screen**

****

**5.6.6 Direction Screen**

public class GetDirections extends Activity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_get\_directions);

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.activity\_get\_directions, menu);

// get action bar

ActionBar actionBar = getActionBar();

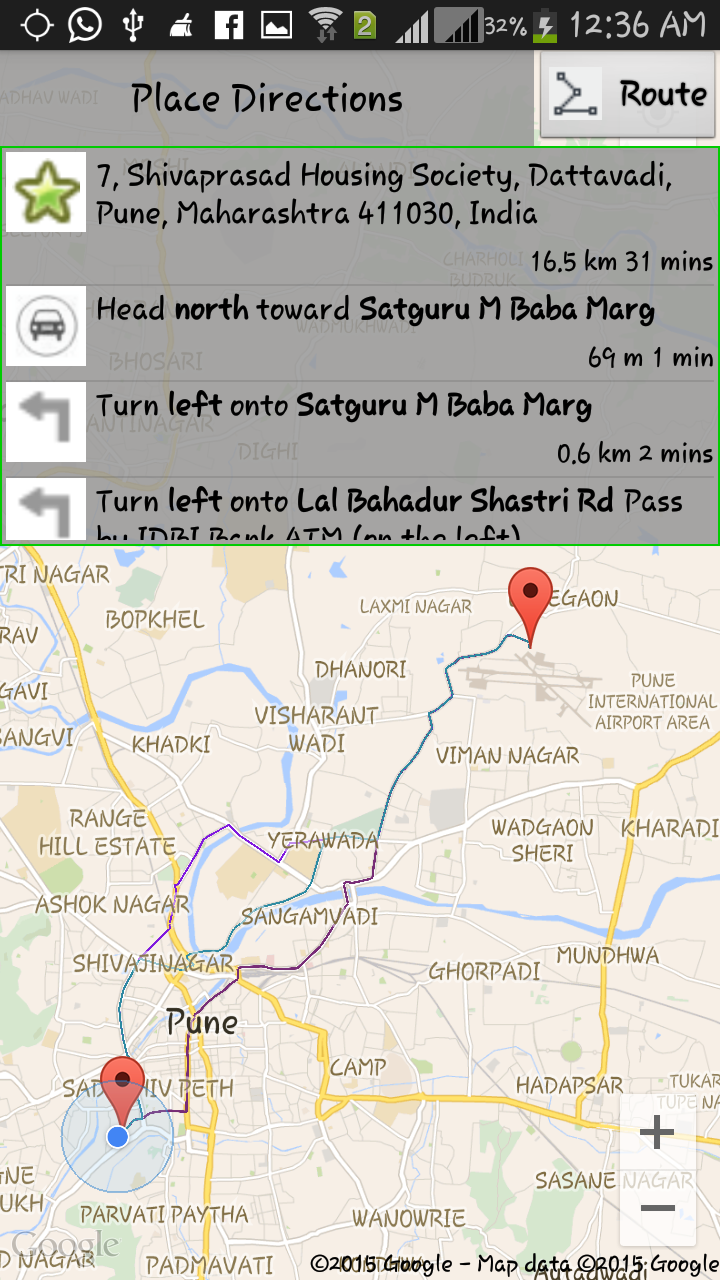
// Enabling Up / Back navigation

actionBar.setDisplayHomeAsUpEnabled(true);

return true;

}

* + 1. **Route Screen**

****

**5.6.7 Route Screen**

* + 1. **Weather Screen**

****

**5.6.8 Weather Screen**

public class TempratureHelper {

public static void main(String[] args) {

getWeatherData("18.5089776", "73.791714");

}

public static HashMap map = new HashMap();

static {

map.put("200", "thunderstorm with light rain #11d.png#Thunderstorm");

map.put("201", "thunderstorm with rain #11d.png#Thunderstorm");

map.put("202", "thunderstorm with heavy rain #11d.png#Thunderstorm");

map.put("210", "light thunderstorm #11d.png#Thunderstorm");

map.put("211", "thunderstorm #11d.png#Thunderstorm");

map.put("212", "heavy thunderstorm #11d.png#Thunderstorm");

map.put("221", "ragged thunderstorm #11d.png#Thunderstorm");

map.put("230", "thunderstorm with light drizzle #11d.png#Thunderstorm");

map.put("231", "thunderstorm with drizzle #11d.png#Thunderstorm");

map.put("232", "thunderstorm with heavy drizzle #11d.png#Thunderstorm");

map.put("300", "light intensity drizzle #09d.png#Drizzle");

map.put("301", "drizzle #09d.png#Drizzle");

map.put("302", "heavy intensity drizzle #09d.png#Drizzle");

map.put("310", "light intensity drizzle rain #09d.png#Drizzle");

map.put("311", "drizzle rain #09d.png#Drizzle");

map.put("312", "heavy intensity drizzle rain #09d.png#Drizzle");

map.put("313", "shower rain and drizzle #09d.png#Drizzle");

map.put("314", "heavy shower rain and drizzle #09d.png#Drizzle");

map.put("321", "shower drizzle #09d.png#Drizzle");

map.put("500", "light rain #10d.png#Rain");

map.put("501", "moderate rain #10d.png#Rain");

map.put("502", "heavy intensity rain #10d.png#Rain");

map.put("503", "very heavy rain #10d.png#Rain");

map.put("504", "extreme rain #10d.png#Rain");

map.put("511", "freezing rain #13d.png#Rain");

map.put("520", "light intensity shower rain #09d.png#Rain");

map.put("521", "shower rain #09d.png#Rain");

map.put("522", "heavy intensity shower rain #09d.png#Rain");

map.put("531", "ragged shower rain #09d.png#Rain");

map.put("600", "light snow #13d.png#Snow");

map.put("601", "snow #13d.png#Snow");

map.put("602", "heavy snow #13d.png#Snow");

map.put("611", "sleet #13d.png#Snow");

map.put("612", "shower sleet #13d.png#Snow");

map.put("615", "light rain and snow #13d.png#Snow");

map.put("616", "rain and snow #13d.png#Snow");

map.put("620", "light shower snow #13d.png#Snow");

map.put("621", "shower snow #13d.png#Snow");

map.put("622", "heavy shower snow #13d.png#Snow");

map.put("701", "mist #50d.png#Atmosphere");

map.put("711", "smoke #50d.png#Atmosphere");

map.put("721", "haze #50d.png#Atmosphere");

map.put("731", "sand, dust whirls #50d.png#Atmosphere");

map.put("741", "fog #50d.png#Atmosphere");

map.put("751", "sand #50d.png#Atmosphere");

map.put("761", "dust #50d.png#Atmosphere");

map.put("762", "volcanic ash #50d.png#Atmosphere");

map.put("771", "squalls #50d.png#Atmosphere");

map.put("781", "tornado #50d.png#Atmosphere");

map.put("800", "clear sky #01d.png#Clouds");

map.put("801", "few clouds #02d.png #Clouds");

map.put("802", "scattered clouds #03d.png#Clouds");

map.put("803", "broken clouds #04d.png #Clouds");

map.put("804", "overcast clouds #04d.png#Clouds");

map.put("900", "tornado#NA#Extreme");

map.put("901", "tropical storm#NA#Extreme");

map.put("902", "hurricane#NA#Extreme");

map.put("903", "cold#NA#Extreme");

map.put("904", "hot#NA#Extreme");

map.put("905", "windy#NA#Extreme");

map.put("906", "hail#NA#Extreme");

map.put("951", "calm#NA#Additional");

map.put("952", "light breeze#NA#Additional");

map.put("953", "gentle breeze#NA#Additional");

map.put("954", "moderate breeze#NA#Additional");

map.put("955", "fresh breeze#NA#Additional");

map.put("956", "strong breeze#NA#Additional");

map.put("957", "high wind, near gale#NA#Additional");

map.put("958", "gale#NA#Additional");

map.put("959", "severe gale#NA#Additional");

map.put("960", "storm#NA#Additional");

map.put("961", "violent storm#NA#Additional");

map.put("962", "hurricane #NA#Additional");

}

public static HashMap getWeatherData(String lat, String lng) {

String q = "http://api.openweathermap.org/data/2.5/weather?lat="+ lat.trim() + "&lon=" + lng.trim();

StringBuffer json = StringHelper.readURLContent(q);

HashMap hm = new HashMap();

try {

JSONObject jobj = new JSONObject(json.toString());

JSONObject temp = (JSONObject) jobj.get("main");

JSONArray weather = (JSONArray) jobj.get("weather");

System.out.println(weather.length());

JSONObject we = (JSONObject) weather.get(0);

System.out.println(we.get("id"));

System.out.println(we.get("main"));

System.out.println(we.get("description"));

System.out.println(StringHelper.n2d(temp.get("temp")));

double temp2 = StringHelper.n2d(temp.get("temp"));

temp2 = temp2 - 273;

hm.put("temp", temp2);

hm.put("id", we.get("id"));

hm.put("main", we.get("main"));

hm.put("description", we.get("description"));

} catch (JSONException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return hm;

}

s = PreferenceManager.getDefaultSharedPreferences(HomeActivity.this);

type = s.getString("types", CategoryConstants.ATTRACTIONS);

distance = s.getString("distance", "1000");

placesHelper = new PlacesHelper();

buttonDistance = (TextView) findViewById(R.id.textView4);

weatherImageView = (ImageView) findViewById(R.id.imageViewWeather);

textViewWeather = (TextView) findViewById(R.id.textViewWeather);

buttonType = (TextView) findViewById(R.id.category);

System.out.println("Home oncreate");

addressStr = this.getIntent().getStringExtra("addr");

System.out.println("addressStr.................................."

+ addressStr);

}

* + 1. **Place Information Screen**

****

**5.6.9 Place Information Screen**

public class Place extends Activity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_place);

placeName = (TextView) findViewById(R.id.textViewPlaceName);

placeAdd = (TextView) findViewById(R.id.textViewPlaceAddress);

placePhoneNumber = (TextView) findViewById(R.id.textViewPhoneNumber);

imageView = (ImageView) findViewById(R.id.serverImageView);

ratingBar = (RatingBar) findViewById(R.id.ratingBar1);

url = (TextView) findViewById(R.id.textViewURL);

clodesMSG = (TextView) findViewById(R.id.textViewclosedMSG);

txtreview = (TextView) findViewById(R.id.textView6);

txtopeningsHRS = (TextView) findViewById(R.id.textView8);

PlaceEntity placeEntity = (PlaceEntity) getIntent()

.getSerializableExtra("place");

placeName.setText(placeEntity.getDetailEntity().getName());

placeAdd.setText(placeEntity.getDetailEntity().getFormatted\_address());

String ph = placeEntity.getDetailEntity().getFormatted\_phone\_number();

if (ph == null || ph.equals("")) {

placePhoneNumber.setText("No Data found");

} else {

placePhoneNumber.setText(ph);

}

String urlstr = placeEntity.getDetailEntity().getUrl();

if (urlstr == null || urlstr.equals("")) {

url.setText("No Data found");

} else {

url.setText(Html.fromHtml("<a href=\"" + urlstr + "\">" + urlstr+ "</a>"));

url.setMovementMethod(LinkMovementMethod.getInstance());

}

boolean open = placeEntity.getDetailEntity().isOpen\_now();

if (!open) {

imageView.setImageResource(R.drawable.closed\_sign);

clodesMSG

.setText("Closed can also mean there is no data for this field !");

} else {

imageView.setImageResource(R.drawable.open\_sign);

clodesMSG.setText("");

}

Integer rating = placeEntity.getDetailEntity().getUser\_ratings();

if (rating == null) {

rating = 0;

}

ratingBar.setEnabled(false);

ratingBar.setStepSize(0.5f);

ratingBar.setNumStars(5);

ratingBar.setMax(5);

ratingBar.setRating(rating);

DayEntity[] dayEntities = placeEntity.getDetailEntity()

.getOpening\_HRS();

String openingHRSData = "";

if (dayEntities != null && dayEntities.length > 0) {

for (DayEntity dayEntity : dayEntities) {

try {

openingHRSData = openingHRSData+ StringHelper.getDay(dayEntity.getDay())

+ " \r\n Open Hrs : " + dayEntity.getOpenHRS()+ " \r\n Close Hrs: " + dayEntity.getCloseHRS();

openingHRSData = openingHRSData+ "\r\n--------------------------------------------------------------------\r\n";

} catch (Exception e) {

}

}

txtopeningsHRS.setText(openingHRSData);

} else if (openingHRSData.equals("")) {

txtopeningsHRS.setText("No Data found");

}

ReviewEntity[] entities = placeEntity.getDetailEntity().getReviews();

String reviewsData = "";

if (entities != null && entities.length > 0) {

for (ReviewEntity entity : entities) {

reviewsData = reviewsData + "Author :" + entity.getAuthorName();

reviewsData = reviewsData + "\r\nReview :" + entity.getText();

reviewsData = reviewsData+ "\r\n--------------------------------------------------------------------\r\n";

}

txtreview.setText(reviewsData);

} else {

txtreview.setText("No Data found");

}

}

public void callNumber(View view) {

String phone\_no = placePhoneNumber.getText().toString().replaceAll("-", "");

Intent callIntent = new Intent(Intent.ACTION\_CALL);

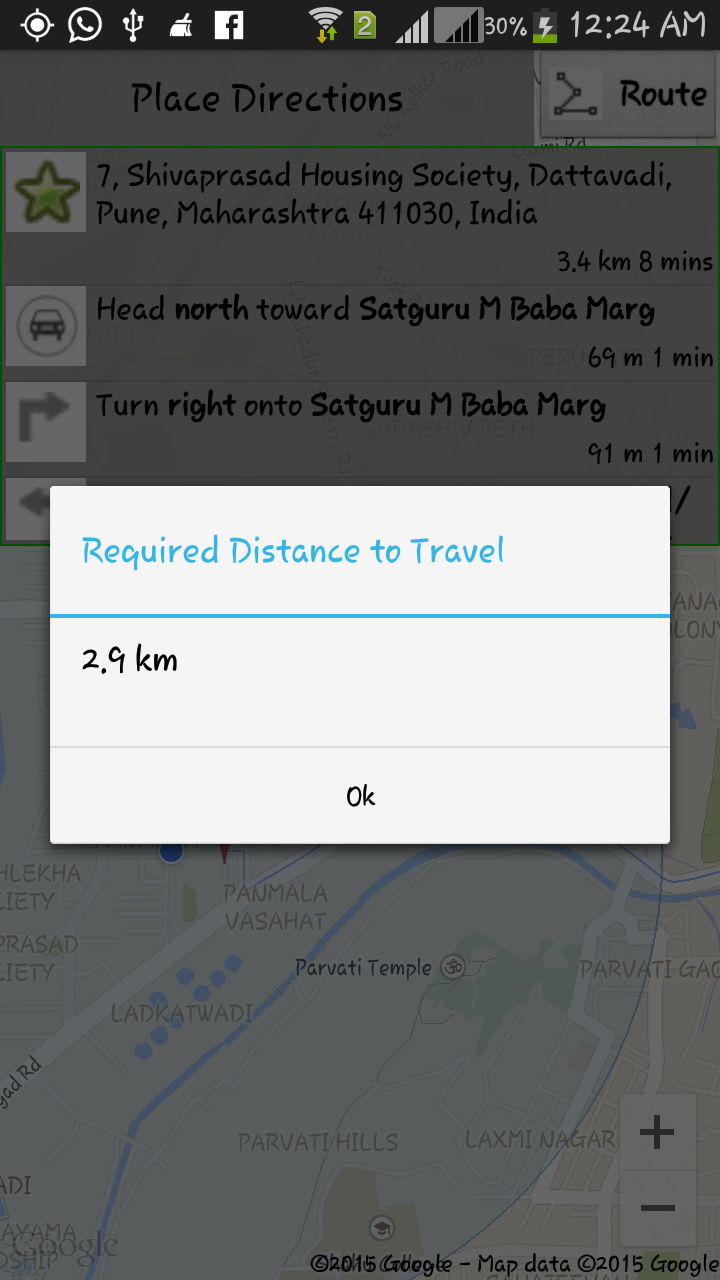
callIntent.setData(Uri.parse("tel:" + phone\_no));

startActivity(callIntent);

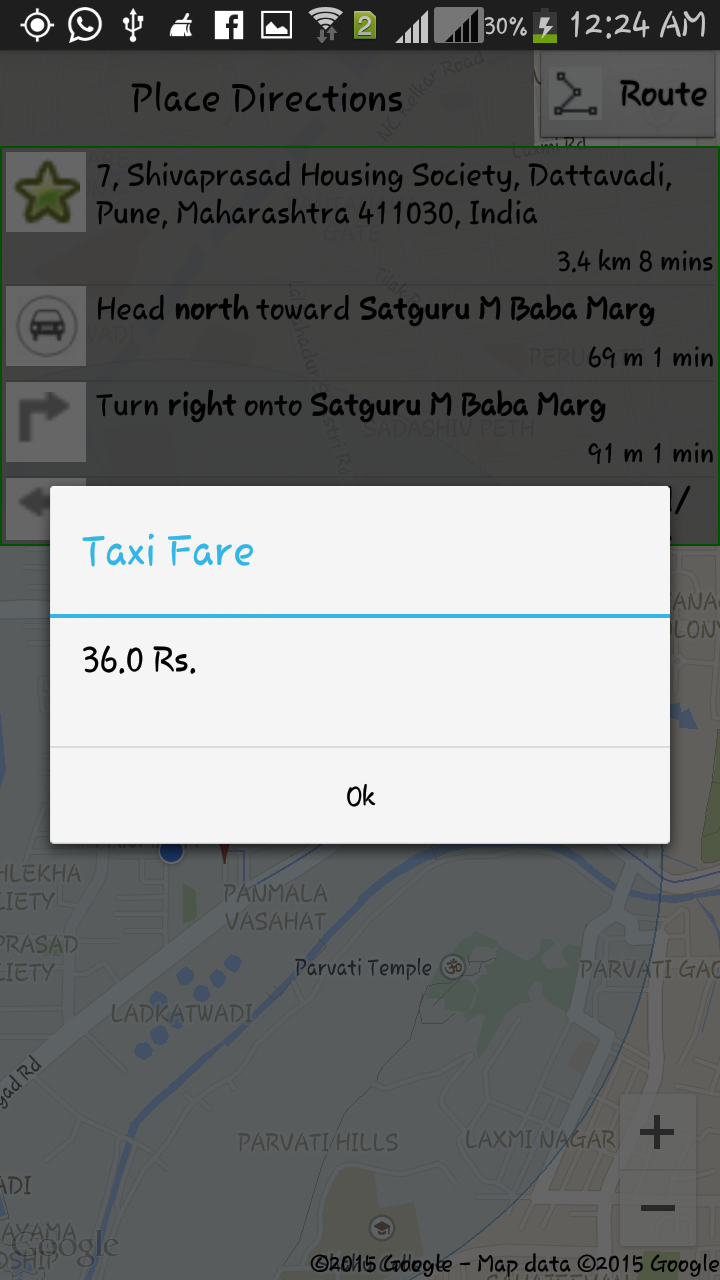
}

}

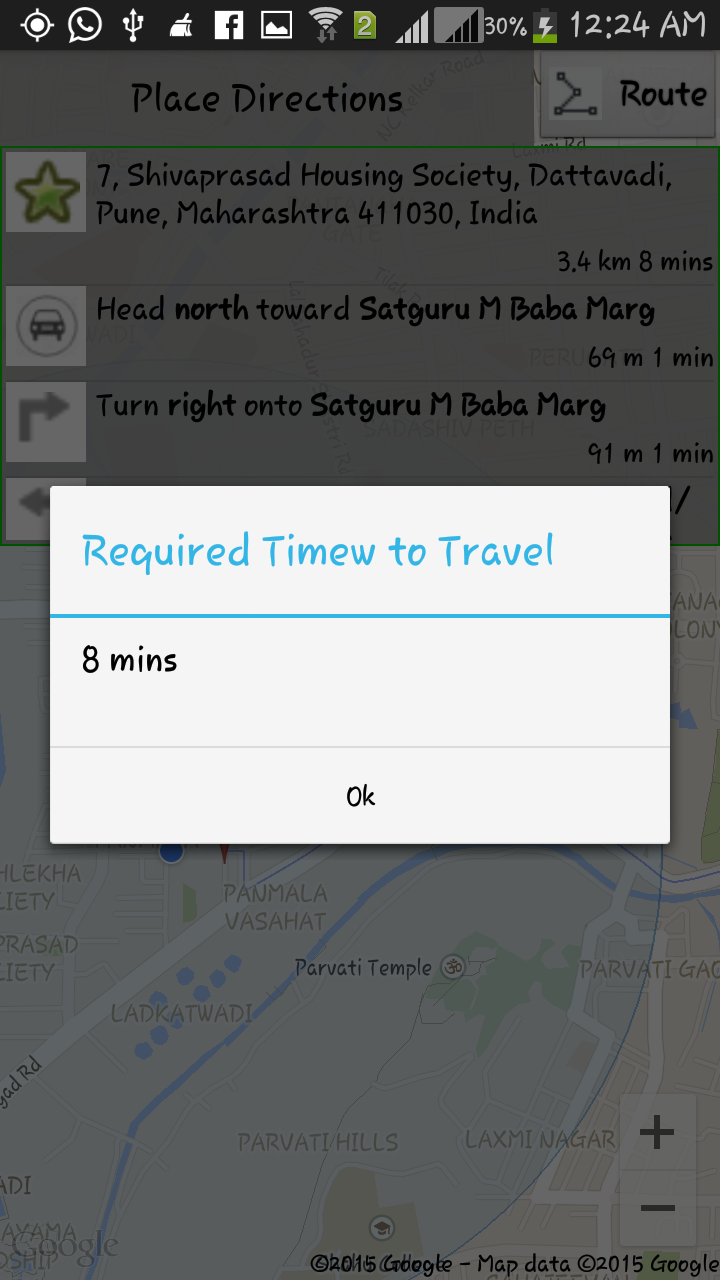
* + 1. **Time, Distance and Fare Screens**

****

**5.6.10.1 Distance Screen**

****

**5.6.10.2 Fare Screen**



**5.6.10.3 Time Screen**

@Override

public boolean onOptionsItemSelected(MenuItem item) {

Intent main = null;

final int timeId = R.id.time;

final int distanceId = R.id.distance;

if (item.getItemId() == timeId) {

showAlert(time, "Required Timew to Travel");

} else if (item.getItemId() == distanceId) {

showAlert(distance, "Required Distance to Travel");

} else if (item.getItemId() == R.id.item1) {

SharedPreferences s=PreferenceManager.getDefaultSharedPreferences(GoogleMapActivity.this);

int rate = Integer.parseInt((s.getString("taxirates", "10")));

int rate1km = Integer

.parseInt((s.getString("taxiRates1stkm", "17")));

double fare = 0;

String distanceStr = distance.substring(0,

distance.indexOf("km") - 1);

double distanceInt = Double.parseDouble(distanceStr);

if (distanceInt > 1) {

fare = (distanceInt - 1) \* rate;

fare = fare + rate1km;

} else {

fare = rate1km;

}

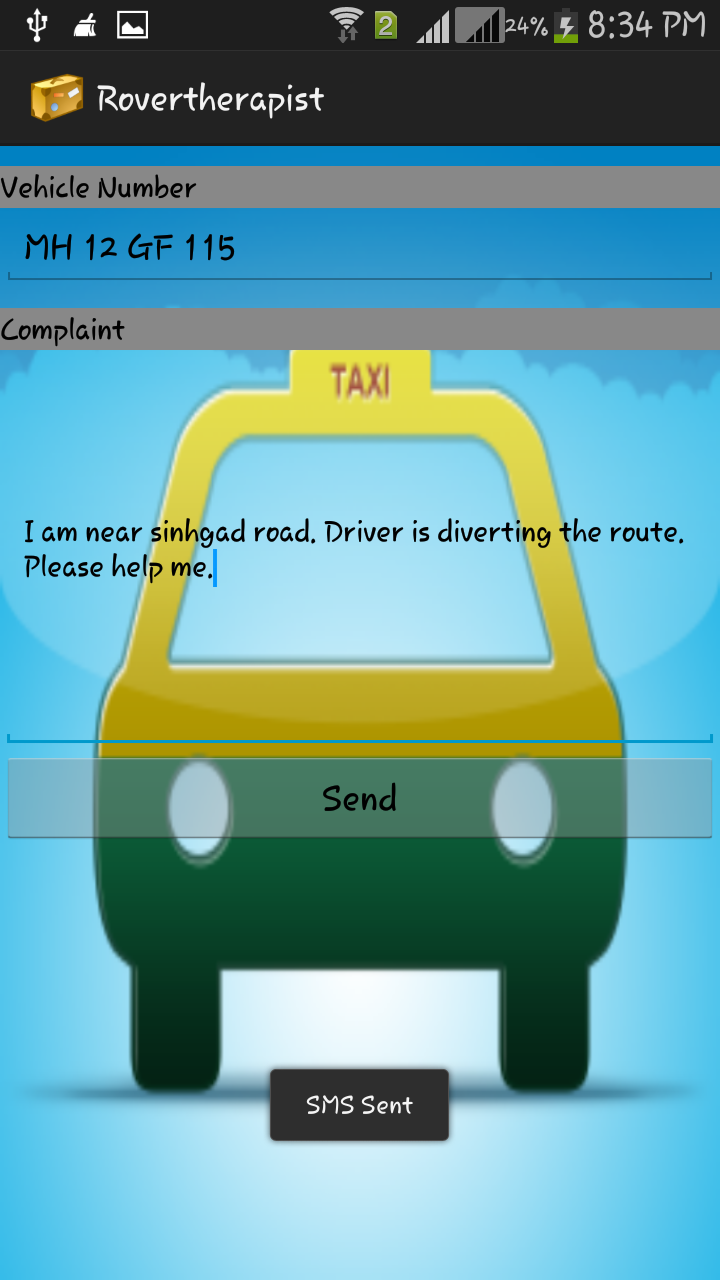
showAlert(String.valueOf(fare) + " Rs. ", "Taxi Fare");

}

return true;

}

* + 1. **Launch Compliant Screen**

****

**5.6.11 Launch Compliant Screen**

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.launchcomplaint);

txtVehiclenumber = (EditText) findViewById(R.id.txtVechicleNumber);

txtcomplaint = (EditText) findViewById(R.id.txtComplaint);

src = getIntent().getStringExtra("src");

dest = getIntent().getStringExtra("dest");

orignalFare = getIntent().getFloatExtra("orignalfare", 0f);

if (src != null && dest != null && orignalFare != null) {

txtcomplaint.append("Start : " + src + "\nDestination : " + dest+ "\nOrginal Fare : " + orignalFare + "\nFare Taken : ");

}

public void sendcomplaint(View view) {

try {

String phoneNo = AndroidConstants.COMPLAINT\_NUMBER;

System.out.println("reading complaint ....");

String complaint = txtcomplaint.getText().toString();

System.out.println("read complaint ...." + complaint);

String vehicleno = txtVehiclenumber.getText().toString();

System.out.println("read veh No ...." + vehicleno);

String message = "Complaint for Vehicle No. " + vehicleno + " :- "+ complaint;

System.out.println("Message to send : " + message);

if (vehicleno.length() > 0 && message.length() > 0

&& message != null && vehicleno != null){

sendSMS(phoneNo, message);

}

else{

Toast.makeText(getBaseContext(),

"Please enter both Vehicle number and complaint.",

Toast.LENGTH\_SHORT).show();

}

} catch (Exception e) {

System.out.println("in exception "

+ txtcomplaint.getText().toString());

Toast.makeText(getApplicationContext(),

"in exception " + e.getMessage(), Toast.LENGTH\_LONG).show();

}

}

private void sendSMS(String phoneNumber, String message) {

String SENT = "SMS\_SENT";

String DELIVERED = "SMS\_DELIVERED";

PendingIntent sentPI = PendingIntent.getBroadcast(this, 0, new Intent(SENT), 0);

PendingIntent deliveredPI = PendingIntent.getBroadcast(this, 0,new Intent(DELIVERED), 0);

// ---when the SMS has been sent---

registerReceiver(new BroadcastReceiver() {

@Override

public void onReceive(Context arg0, Intent arg1) {

switch (getResultCode()) {

case 1: Activity.RESULT\_OK:Toast.makeText(getBaseContext(), "SMS sent",

Toast.LENGTH\_SHORT).show();

break;

case 2: SmsManager.RESULT\_ERROR\_GENERIC\_FAILURE:

Toast.makeText(getBaseContext(), "Generic failure",Toast.LENGTH\_SHORT).show();

break;

case 3: SmsManager.RESULT\_ERROR\_NO\_SERVICE:Toast.makeText(getBaseContext(), "No service",Toast.LENGTH\_SHORT).show();

break;

case 4: SmsManager.RESULT\_ERROR\_NULL\_PDU:Toast.makeText(getBaseContext(), "Null PDU",Toast.LENGTH\_SHORT).show();

break;

case 5: SmsManager.RESULT\_ERROR\_RADIO\_OFF:Toast.makeText(getBaseContext(), "Radio off",Toast.LENGTH\_SHORT).show();

break;

}

}

}, new IntentFilter(SENT));

// ---when the SMS has been delivered---

registerReceiver(new BroadcastReceiver() {

@Override

public void onReceive(Context arg0, Intent arg1) {

switch (getResultCode()) {

case 1: Activity.RESULT\_OK:Toast.makeText(getBaseContext(), "SMS delivered",

Toast.LENGTH\_SHORT).show();

break;

case 2: Activity.RESULT\_CANCELED:Toast.makeText(getBaseContext(), "SMS not delivered",Toast.LENGTH\_SHORT).show();

break;

}

}

}, new IntentFilter(DELIVERED));

SmsManager sms = SmsManager.getDefault();

ArrayList<String> parts = sms.divideMessage(message);

sms.sendMultipartTextMessage(phoneNumber, null, parts, null, null);

Toast.makeText(getApplicationContext(), "SMS Sent", Toast.LENGTH\_LONG)

.show();

}

}

1. **TESTING**
   1. **TESTCASES**
      1. **Manual Testing**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test case id** | **Test case name** | **test case desc** | **test steps** | | | **test status** | **test prority** | **defect severity** |
| **Step** | **Expected** | **actual** |
| User | FutureTouristCRM | Show tourist places | Launch Application. Retrieve Source Location. | Source Location Retrieved Successfully. | Source Location Retrieved Successfully. | Pass | High | Low |
| Enter final Destination. | Menu List View. |  | P | High | Low |
|  |  |  | Highlight Roadmap | Calculate Fare |  | P | high | Low |
| Lodge the complaint against driver. | Complaint Lodge Successfully to nearest RTO. |  | P | high | Low |
| Application | Update Places | To Update Places | Update the Details On Web | Updating is successful. |  | P | Low | low |
|  |  | Gather Information. | Gather Information from diff Sources | Successful. Sent to Client. |  | P | Low | Low |
|  |  | Calculate Fare. | Update sent to Client. | Successful. |  | P | Low | Low |
| GPS | Position Detector | To Detect Position | Detect client position. | Successful. |  |  |  |  |

**6.1.1 Test Case 1**

**Android Application**

1. **Splash Screen**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test case id** | **Test case name** | **test case desc** | **test steps**  **Step Actual Excepted** | **test status** | **test prority** | **defect severity** |
| 1.1 | Splash Screens | Auto Sliding of the screens | Open the Splash  Applicati screen  On should  appear | pass | low | low |

1. **sign in as guest**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test case id** | **Test case name** | **test case desc** | **test steps** | | | **test status** | **test prority** | **defect severity** |
| **Step** | **Expected** | **actual** |
| 2.1 | Sign in as guest | Sign in as guest message | Sign in as guest message->Yes | Register user screen should appear | Register user screen appears. | Pass | High | Low |
| Sign in as guest message->No | Server Screen Should appear | Server screen appears | Pass | High | Low |

1. **Register user screen**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test case id** | **Test case name** | **test case desc** | **test steps** | | | **test status** | **test prority** | **defect severity** |
| **Step** | **Expected** | **actual** |
| 3.1 |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |

1. **Server connection parameter Screen**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test case id** | **Test case name** | **test case desc** | **test steps** | | | **test status** | **test prority** | **defect severity** |
| **Step** | **Expected** | **actual** |
| 4.1 |  | Textfields ip, port, rate is touched | Touch the textfields | Keyboard should appear | Keboard appears. | Pass | High | Low |
| 4.2 |  | Connectivity button is touched | Touch the connectivity button | Connection to server should be established | Connection is established | Pass | high | Low |

1. **Main Screen**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test case id** | **Test case name** | **test case desc** | **test steps** | | | **test status** | **test prority** | **defect severity** |
| **Step** | **Expected** | **actual** |
| 5.1 |  | Menu is touched | Touch the menu | Dropdown list should appear | Dropdown list appears. | Pass | High | Low |
| 5.2 |  | Current location of user |  | Current location of user should be displayed | Current location of user is displayed | Pass | high | Low |
| 5.3 |  | Direction symbol is touched | Touch the direction symbol | User should be navigated to the direction screen | User is navigated to the direction screen | Pass | high | low |
| 5.4 |  | Information symbol is touched | Touch the information symbol | User should be navigated to the information screen | User is navigated to the information screen | Pass | high | low |
| 5.5 |  | Location symbol is touched | Touch the location symbol | User should be navigated to the location screen | User is navigated to the location screen | Pass | high | low |

1. **Menu Screen**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test case id** | **Test case name** | **test case desc** | **test steps** | | | **test status** | **test prority** | **defect severity** |
| **Step** | **Expected** | **actual** |
| 6.1 |  | Change distance is touched | Touch the change distance option | Distances should appear and as per the distance selected by the user, list of places should appear. | Distances appears and as per the distance selected by the user, list of places appears.  . | Pass | High | Low |
| 6.2 |  | Change category is touched | Touch the change category option | User should be navigated to the list of attractions screen and as per the attractions selected by the user places on the main screen should appear | User is navigated to the list of attractions screen and as per the attractions selected by the user places on the main screen appears | Pass | high | Low |
| 6.3 |  | Launch complaint is touched | Touch the launch compliant option | User should be navigated to the complaint screen | User is navigated to complaint screen | Pass | high | low |
| 6.4 |  | Display of distance and fare calculated | Touch the information symbol | User should be able to get a popup of the calculated disatnce and fare | User gets the calculated distance and fare | Pass | high | low |

**WEB MODULE**

* + 1. **Automated Testing**

1. **CONCLUSION AND FUTURE ENHANCEMENT**