**CHAPTER 1**

**INTRODUCTION**

**1.1 OVERVIEW**

As the population of the country increases, the medical emergencies are also increases. Today there are hundreds of Ambulance services available but its hectic for a person to contact the nearest ambulance for his/her need in emergency situations. Nowadays mobile apps are widely used and so the ambulance tracking can be done through a mobile app. This application can be used to track the nearest ambulance and contact the driver. For this need two login are created, one for user and other for ambulance driver. In this application, the Google map API is utilized in order to find the location of the nearest ambulance service. ­­­This application will be really useful in times of need.

**1.2 PROBLEM STATEMENT**

Ambulance tracing of 108 service like that of OLA through App, so that nearby persons can plan whether to wait for ambulance or take the patient to thehospital through his own resource in emergency, can track the nearest ambulance before booking and can contact the driver allotted.

**1.3 EXISTING SYSTEM**

In the existing system, once the user on clicks the Emergency App, the application will fetch the user's current location and point. The user can then call helpline by clicking on the Call button. On calling, user’s current position along with user details will be sent to the call center and agents could view the user’s current position in Google maps and can dispatch the nearest ambulance.

**1.3.1 DISADVANTAGES OF EXISTING SYSTEM**

* The existing system does not provide the ambulance location to the user.
* It only trace the user’s location alone and dispatch it to the call center.
* From the call center the agents will send an alert message to the nearest ambulance which will take time for the ambulance driver to reach the user’s location.
* The existing system does not get the secondary confirmation from the user. This will make a call immediately once when the user log in to the application.

**1.4 PROPOSED SYSTEM**

The proposed System focus to make the ambulance available to a needy patient in the shortest possible time by tracking the nearest ambulance from a user’s location and provide a direct interaction between the user and ambulance driver. Pinpointing vehicle locations to determine which ambulance is closest to an emergency call providing real-time directions to ambulance drivers so they can take quick routes while avoiding heavy traffic and accidents. Nowadays mobile apps are widely used and so the ambulance tracking can be done through a mobile app. This application can be used to track the nearest ambulance and contact the driver. For this need two login so an Android app is created, one for user and other for ambulance driver. In this application, the Google map API is utilized in order to find the location of the nearest ambulance service. ­­­This application will be really useful in times of need.

**1.4.1** **ADVANTAGES OF THE PROPOSED SYSTEM**

* Live tracking and status information of all available ambulances in a map which helps for quick redirection of the ambulance to a critical spot.
* Ask secondary confirmation from the user before making a call to the nearest ambulance driver.
* It does not involve any call center as an interface between the user and ambulance driver, so that it will reduce the time to reach the patient’s location.
* Proposed system provide direct interaction between the user and the ambulance driver.

**CHAPTER 2**

**REQUIREMENTS**

**2.1 TOOLS REQUIREMENTS**

**2.1.1 TOOLS USED**

* Front End : Android Studio, Java, XML.
* Back End : Firebase
* Operating System : Windows7/8/10

**2.1.2 MINIMUM REQUIREMENTS OF THE SYSTEM**

* Processor : Intel core i3
* RAM : 4 GB
* HDD : 64 GB
* Monitor : 15’’colour monitor

**2.2 FUNCTIONAL REQUIREMENTS**

* Function to register the user details such as Username, Phone number, mail ID, password.
* Function to login to the user profile with correct mail ID and password.
* Function that allow user to view the map that display nearest location of the ambulance driver with their contact phone number.
* Function that facilitate the user to select a nearest ambulance driver and make a call to him.
* Function to login to the driver profile with correct mail ID and password.
* Function that facilitate driver to view the user’s location once while the driver attend the phone call of that particular user.

**2.3 NON-FUNCTIONAL REQUIREMENTS**

**2.3.1 Performance**

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the staffs of the existing system to give the requirement specifications because they are the people who finally use the system.

This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use. The requirement specification for any system can be broadly stated as given below:

* The Android Mobile should be able to interface with the server
* The Android Mobile should be accurate
* The Android Mobile should be better than the existing system.

**2.3.2 Safety and Security Requirements**

* **User Identification:** The system requires the user to identify himself/herself
* **Login ID**: The user and driver who uses the system shall have a Logon ID and Password.
* **Modification**: Any modification (insert, delete (or) update) for the Database shall besynchronized and done only by the administrator in the ward.

**CHAPTER 3**

**MODULES DESCRIPTION**

**3.1 USER INTERFACE MODULE**

In this module we deal with login interface, storage in and retrieval of data from database in the server. Accordingly, depending on whether the user is authenticated, the interface begins. If the user does not have the login Id before, he/she can register with their details by creating a new account. If an essential field is not filled in the form then an error is thrown stating to fill in that field.

After authentication the user must be able to view current location of the nearest ambulance along with the ambulance driver’s contact number. On clicking the contact number User must be facilitate to make a call to the nearest ambulance driver so that user’s current position along with user details will be sent to that particular driver directly.

**3.2 DRIVER FEATURES**

* Driver have their respective user id & password for authenticated login
* Driver can accept or decline the call of the user based on his convenience.
* Once the driver accept the call, the current user’s location is shared to the driver directly with his details.
* After tracking the user’s location the drive can dispatch to the critical point.

**3.3 PRE-CONDITION**

* The driver should be authenticated.
* They should have their user id and password individually
  1. **POST-CONDITION**
* Login is successful.

**3.5 USERS VIEW:**

* **Locate departure point:**
  + - The user will select the departure point where the accident occurred.
* **Getting location of nearby ambulance:**
  + - User will now see all the ambulance locations that are nearby from which the user will locate the nearest ambulance and sends indication to the ambulance driver.
* **Receiving response:**
  + - User will receive the driver’s response if he accepts the indication, if he rejects due to some reasons users need to do the location process once again.

**3.6 DRIVERS VIEW:**

* **Receiving User request:**
  + - Driver will receive a notification or a call from the user along with the location where the accident has occurred.
* **Sending confirmation:**
  + - The driver need to confirm his acceptance of the request if he is not ready then he needs to reject the request.
* **Navigate to place:**
  + - After accepting the request he navigates to the needy place using the location shared by the user.

**CHAPTER 4**

**DESIGN**

# **4.1 USER INTERFACE MODEL**

**4.1.1 User Login**

* Once the user creates an account he/she can login to the application by entering a correct email ID and password to navigate to the further page.
* If the email ID or password is Invalid it will throw an error message like ‘Authentication Failed’.

# **4.1.2 LOCATION TRACKING:**

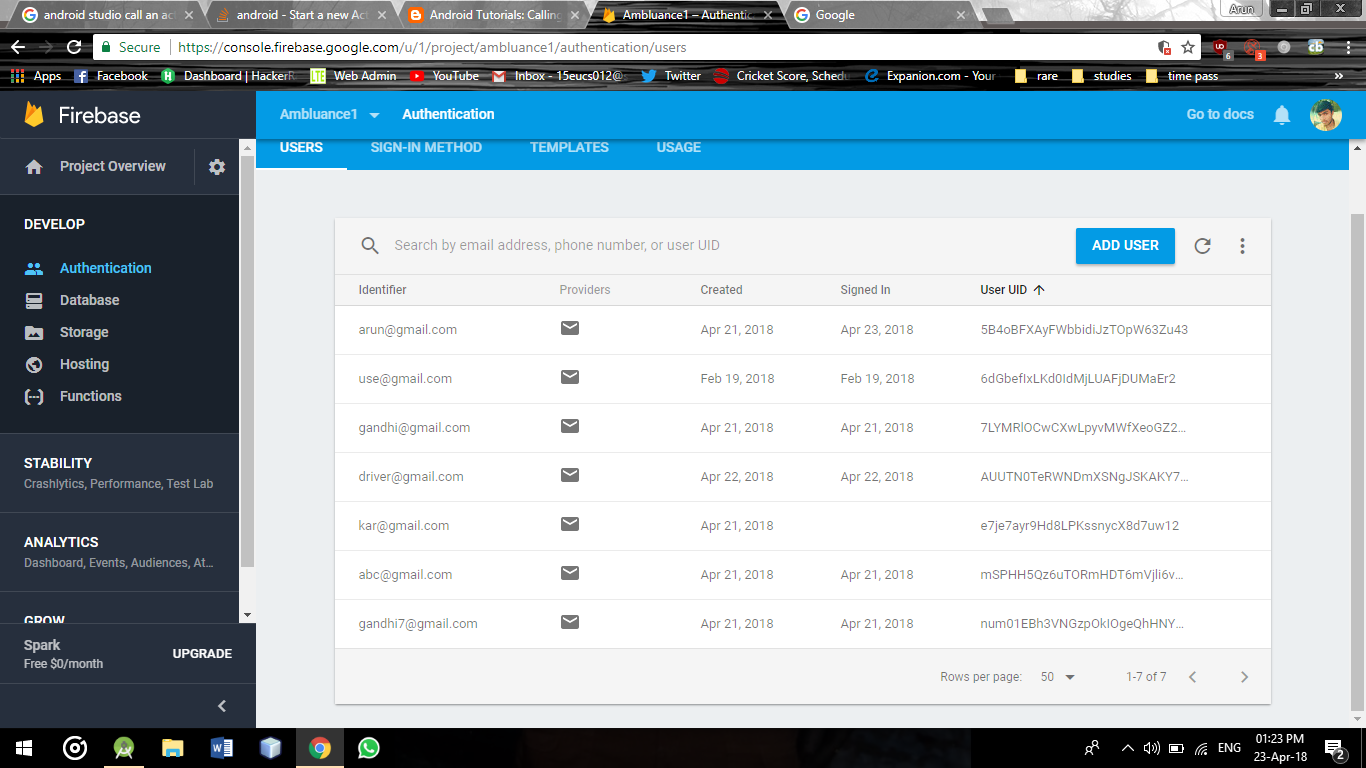
* Once the user login in successfully he/she will navigate to the map layout.
* Here user can track the current position of all available ambulances in a map which helps for quick redirection of the ambulance to a critical spot

**4.2 DATABASE SCHEMA**

* This database schema consists of database named ‘AMBULANCE’ and have a user table.
* These tables have the fields specified in the related forms.

**4.2.1 USER TABLE**

* It consists of list of users who had registered in the application.
* This table consists of Identifier the email id of the user.
* The time when it is created.
* The time when recently signed in.
* The unique UID of each User.



**Figure 4.1 User Table**

**Table 4.1 Structure of User Table**

|  |  |  |
| --- | --- | --- |
| **FIELD NAME** | **DATATYPE** | **DESCRIPTION** |
| Identifier | VARCHAR | User mail ID |
| Created | DATE | Date on which user Create an account |
| Signed in | DATE | Date on which user signed in |
| User UID | VARCHAR | URL ID of the user account. |

**4.3 ARCHITECTURAL FLOW DIAGRAM**

A data flow diagram is a graphical representation of the "flow" of data through an information system. A data flow diagram can also be used for the visualization of data processing (structured design). It is common practice for a designer to draw a context-level DFD first which shows the interaction between the system and outside entities. This context-level DFD is then "exploded" to show more detail of the system being modeled.

**4.3.1 USER FLOW DIAGRAM**

Select Departure point

Get information of all nearby ambulances

No

Send request to nearest ambulance driver

Wait for driver’s response

Yes

Get driver’s acceptance notification

**Figure. 4.3 User flow Diagram**

**4.3.2 DRIVER FLOW DIAGRAM**

**DRIVER**

Receives user request

Sends response to user

Navigate to the place

**Figure.4.4 Driver flow diagram**

**CHAPTER 5**

**IMPLEMENTATION**

**5.1 FRONT END**

* Java – Android App(Versions 2.3 or higher)
* Front end is designed by XML.

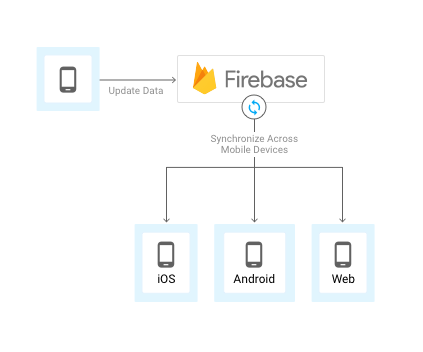
**FEATURES**

* Applications are developed in the Java language using the Android software development kit (SDK).
* The SDK includes a comprehensive set of development tools, including a debugger, software libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials.
* The officially supported integrated development environment (IDE) is Eclipse using the Android Development Tools (ADT) plugin.
* Other development tools are available, including a Native Development Kit for applications or extensions in C or C++, Google App Inventor, a visual environment for novice programmers, and various cross platform mobile web applications frameworks.
* It include tools that interface with Android platform ,such as adb ,fastboot, and systrace.
* These tools are required for Android app development. They are also needed if you want to unlock your device boot loader and flash it with a new system image.

**5.2 BACK END**

### FIREBASE

[Firebase](https://firebase.google.com/) is a fully managed platform for building iOS, Android, and web apps that provides automatic data synchronization, authentication services, messaging, file storage, analytics, and more. Starting with Firebase is an efficient way to build or prototype mobile backend services.



**Figure 5.1 Firebase**

**Recommended for:**

* Limiting on-device data storage by storing JSON data in the Firebase Realtime Database and files in Firebase Storage.
* Sending notifications with Firebase Cloud Messaging.
* Automated real-time data synchronization across multiple devices.
* Gracefully handling the offline case.
* Authenticating users through a variety of identity providers.
* Rapid development of a backend service.

**Not recommended for:**

* Apps that need a backend service to modify the synchronized data.

**5.3 CODING FOR THE APPLICATION**

**5.3.1. XML CODE**

**Register Activity:**

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

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

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

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context="com.example.welcome.ambluance1.Register">

<EditText

android:id="@+id/reg\_name"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_alignParentStart="true"

android:layout\_below="@+id/editText4"

android:layout\_marginTop="13dp"

android:ems="10"

android:hint="mail id"

android:inputType="textPersonName"

android:textColor="@android:color/black" />

<EditText

android:id="@+id/reg\_password"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_alignParentStart="true"

android:layout\_below="@+id/reg\_name"

android:layout\_marginTop="13dp"

android:ems="10"

android:hint="Password"

android:inputType="textPassword" />

<Button

android:id="@+id/register"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_alignParentBottom="true"

android:layout\_centerHorizontal="true"

android:layout\_marginBottom="86dp"

android:text="REGISTER"

android:textColor="#FFFFFF"

android:textSize="25sp" />

<EditText

android:id="@+id/editText3"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_alignParentStart="true"

android:layout\_alignParentTop="true"

android:ems="10"

android:hint="Name"

android:inputType="textPersonName" />

<EditText

android:id="@+id/editText4"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_alignParentStart="true"

android:layout\_below="@+id/editText3"

android:layout\_marginTop="11dp"

android:ems="10"

android:hint="Phone number"

android:inputType="number" />

</RelativeLayout>

**Main Activity:**

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

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

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

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context="com.example.welcome.ambluance1.MainActivity">

<Button

android:id="@+id/user"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentTop="true"

android:layout\_centerHorizontal="true"

android:layout\_marginTop="68dp"

android:text="user"

android:onClick="userlogin"/>

<Button

android:id="@+id/driver"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignStart="@+id/user"

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

android:layout\_marginTop="41dp"

android:text="driver"

android:onClick="driverlogin"/>

</android.widget.RelativeLayout>

**Maps Activity:**

<FrameLayout

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

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

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

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

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

android:id="@+id/map"

android:name="com.google.android.gms.maps.SupportMapFragment"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context="com.example.welcome.ambluance1.MapsActivity" />

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

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

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="vertical">

<Button

android:id="@+id/btnHospital"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Nearby Ambulances" />

<Button

android:id="@+id/btnIns"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Logout" />

</LinearLayout>

</FrameLayout>

**5.3.2 JAVA CODE**

**Register Activity:**

package com.example.welcome.ambluance1;

import …

public class Register extends AppCompatActivity implements View.OnClickListener{

EditText name,phone,mail\_id,password;

Button register;

public FirebaseAuth firebaseAuth;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_register);

mail\_id=(EditText)findViewById(R.id.reg\_name);

password=(EditText)findViewById(R.id.reg\_password);

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

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

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

register.setOnClickListener(this);

firebaseAuth= FirebaseAuth.getInstance();

if(firebaseAuth.getCurrentUser()!=null)

{

finish();

startActivity(new Intent(getApplicationContext(),MainScreen.class));

}

}

public void Regis()

{

String str\_name,str\_password;

str\_name=mail\_id.getText().toString();

str\_password=password.getText().toString();

if(TextUtils.isEmpty(str\_name) || TextUtils.isEmpty(str\_password))

{

Toast.makeText(this,"Fields are Empty!!",Toast.LENGTH\_LONG).show();

return;

}

firebaseAuth.createUserWithEmailAndPassword(str\_name,str\_password)

.addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {

@Override

public void onComplete(@NonNull Task<AuthResult> task) {

if (task.isSuccessful()) {

Toast.makeText(Register.this, "Sucessfuly Registered!", Toast.LENGTH\_LONG).show();

finish();

startActivity(new Intent(getApplicationContext(), UserLogin.class));

} else {

String TAG="AAA";

Log.w(TAG, "createUserWithEmail:failure", task.getException());

Toast.makeText(Register.this, "Failed to Register!...Please Try Again!", Toast.LENGTH\_LONG).show();

}

}

});

}

@Override

public void onClick(View view)

{

if(view==register)

{

Regis();

}

}

}

**Main Actitvity:**

package com.example.welcome.ambluance1;

import …

public class MainActivity extends AppCompatActivity {

private FirebaseAuth firebaseAuth;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

firebaseAuth= FirebaseAuth.getInstance();

FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();

if (user != null)

{

String name = user.getUid();

if(name.contains("driver"))

{

finish();

startActivity(new Intent(getApplicationContext(),MainScreen.class));

}

else

{

finish();

startActivity(new Intent(getApplicationContext(),MapsActivity.class));

}

}

}

public void userlogin(View view){

Intent intent = new Intent(this,UserLogin.class);

startActivity(intent);

}

public void driverlogin(View view){

Intent intent = new Intent(this,DriverLogin.class);

startActivity(intent);

}

}

**Maps Activity:**

package com.example.welcome.ambluance1;

import …

public class MapsActivity extends FragmentActivity implements OnMapReadyCallback,

GoogleApiClient.ConnectionCallbacks,

GoogleApiClient.OnConnectionFailedListener,

LocationListener {

public static LatLng llatLng;

private GoogleMap mMap;

double latitude;

double longitude;

private int PROXIMITY\_RADIUS = 1000;

GoogleApiClient mGoogleApiClient;

Location mLastLocation;

Marker mCurrLocationMarker;

LocationRequest mLocationRequest;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_maps);

Button logout=(Button)findViewById(R.id.btnIns);

logout.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

FirebaseAuth firebaseAuth= FirebaseAuth.getInstance();

firebaseAuth.signOut();

finish();

startActivity(new Intent(getApplicationContext(),UserLogin.class));

}

});

if (android.os.Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.M) {

checkLocationPermission();

}

//Check if Google Play Services Available or not

if (!CheckGooglePlayServices()) {

Log.d("onCreate", "Finishing test case since Google Play Services are not available");

finish();

}

else {

Log.d("onCreate","Google Play Services available.");

}

SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()

.findFragmentById(R.id.map);

mapFragment.getMapAsync(this);

}

private boolean CheckGooglePlayServices() {

GoogleApiAvailability googleAPI = GoogleApiAvailability.getInstance();

int result = googleAPI.isGooglePlayServicesAvailable(this);

if(result != ConnectionResult.SUCCESS) {

if(googleAPI.isUserResolvableError(result)) {

googleAPI.getErrorDialog(this, result,

0).show();

}

return false;

}

return true;

}

@Override

public void onMapReady(GoogleMap googleMap) {

mMap = googleMap;

mMap.setMapType(GoogleMap.MAP\_TYPE\_NORMAL);

//Initialize Google Play Services

if (android.os.Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.M) {

if (ContextCompat.checkSelfPermission(this,

Manifest.permission.ACCESS\_FINE\_LOCATION)

== PackageManager.PERMISSION\_GRANTED) {

buildGoogleApiClient();

mMap.setMyLocationEnabled(true);

}

} else {

buildGoogleApiClient();

mMap.setMyLocationEnabled(true);

}

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

btnHospital.setOnClickListener(new View.OnClickListener() {

String Hospital = "hospital";

@Override

public void onClick(View v) {

Log.d("onClick", "Button is Clicked");

mMap.clear();

String url = getUrl(latitude, longitude, Hospital);

Object[] DataTransfer = new Object[2];

DataTransfer[0] = mMap;

DataTransfer[1] = url;

Log.d("onClick", url);

GetNearbyPlacesData getNearbyPlacesData = new GetNearbyPlacesData(MapsActivity.this);

getNearbyPlacesData.execute(DataTransfer);

Toast.makeText(MapsActivity.this, "Nearby Ambulances", Toast.LENGTH\_LONG).show();

}

});

}

protected synchronized void buildGoogleApiClient() {

mGoogleApiClient = new GoogleApiClient.Builder(this)

.addConnectionCallbacks(this)

.addOnConnectionFailedListener(this)

.addApi(LocationServices.API)

.build();

mGoogleApiClient.connect();

}

@Override

public void onConnected(Bundle bundle) {

mLocationRequest = new LocationRequest();

mLocationRequest.setInterval(1000);

mLocationRequest.setFastestInterval(1000);

mLocationRequest.setPriority(LocationRequest.PRIORITY\_BALANCED\_POWER\_ACCURACY);

if (ContextCompat.checkSelfPermission(this,

Manifest.permission.ACCESS\_FINE\_LOCATION)

== PackageManager.PERMISSION\_GRANTED) {

LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient, mLocationRequest, this);

}

}

private String getUrl(double latitude, double longitude, String nearbyPlace) {

StringBuilder googlePlacesUrl = new StringBuilder("https://maps.googleapis.com/maps/api/place/nearbysearch/json?");

googlePlacesUrl.append("location=" + latitude + "," + longitude);

googlePlacesUrl.append("&radius=" + PROXIMITY\_RADIUS);

googlePlacesUrl.append("&type=" + nearbyPlace);

googlePlacesUrl.append("&sensor=true");

googlePlacesUrl.append("&key=" + "AIzaSyATuUiZUkEc\_UgHuqsBJa1oqaODI-3mLs0");

Log.d("getUrl", googlePlacesUrl.toString());

return (googlePlacesUrl.toString());

}

@Override

public void onConnectionSuspended(int i) {

}

@Override

public void onLocationChanged(Location location) {

Log.d("onLocationChanged", "entered");

mLastLocation = location;

if (mCurrLocationMarker != null) {

mCurrLocationMarker.remove();

}

latitude = location.getLatitude();

longitude = location.getLongitude();

llatLng = new LatLng(location.getLatitude(), location.getLongitude());

MarkerOptions markerOptions = new MarkerOptions();

markerOptions.position(llatLng);

markerOptions.title("Current-Position"); markerOptions.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE\_MAGENTA));

mCurrLocationMarker = mMap.addMarker(markerOptions);

mMap.moveCamera(CameraUpdateFactory.newLatLng(llatLng));

mMap.animateCamera(CameraUpdateFactory.zoomTo(16));

Toast.makeText(MapsActivity.this,"Your Current Location", Toast.LENGTH\_LONG).show();

Log.d("onLocationChanged", String.format("latitude:%.3f longitude:%.3f",latitude,longitude));

if (mGoogleApiClient != null) {

LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient, this);

Log.d("onLocationChanged", "Removing Location Updates");

}

Log.d("onLocationChanged", "Exit");

}

@Override

public void onConnectionFailed(ConnectionResult connectionResult) {

}

public static final int MY\_PERMISSIONS\_REQUEST\_LOCATION = 99;

public boolean checkLocationPermission(){

if (ContextCompat.checkSelfPermission(this,

Manifest.permission.ACCESS\_FINE\_LOCATION)

!= PackageManager.PERMISSION\_GRANTED) {

if (ActivityCompat.shouldShowRequestPermissionRationale(this,

Manifest.permission.ACCESS\_FINE\_LOCATION)) {

ActivityCompat.requestPermissions(this,

new String[]{Manifest.permission.ACCESS\_FINE\_LOCATION},

MY\_PERMISSIONS\_REQUEST\_LOCATION);

} else {

ActivityCompat.requestPermissions(this,

new String[]{Manifest.permission.ACCESS\_FINE\_LOCATION},

MY\_PERMISSIONS\_REQUEST\_LOCATION);

}

return false;

} else {

return true;

}

}

@Override

public void onRequestPermissionsResult(int requestCode,

String permissions[], int[] grantResults) {

switch (requestCode) {

case MY\_PERMISSIONS\_REQUEST\_LOCATION: {

if (grantResults.length > 0

&& grantResults[0] == PackageManager.PERMISSION\_GRANTED) {

if (ContextCompat.checkSelfPermission(this,

Manifest.permission.ACCESS\_FINE\_LOCATION)

== PackageManager.PERMISSION\_GRANTED) {

if (mGoogleApiClient == null) {

buildGoogleApiClient();

}

mMap.setMyLocationEnabled(true);

}

} else {

Toast.makeText(this, "permission denied", Toast.LENGTH\_LONG).show();

}

return;

}

}

}

public LatLng get()

{

return llatLng;

}

}

**CHAPTER 6**

**TESTING**

**6.1 INTRODUCTION**

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. In fact, testing is the one step in the software engineering process that could be viewed as destructive rather than constructive. A strategy for software testing integrates software test case design methods into a well-planned series of steps that result in the successful construction of software. Testing is the set of activities that can be planned in advance and conducted systematically. The underlying motivation of program testing is to affirm software quality with methods that can economically and effectively apply to both strategic to both large and small-scale systems.

**6.2 STRATEGIC APPROACH TO SOFTWARE TESTING**

The software engineering process can be viewed as a spiral. Initially system engineering defines the role of software and leads to software requirement analysis where the information domain, functions, behaviour, performance, constraints and validation criteria for software are established. Moving inward along the spiral, we come to design and finally to coding. To develop computer software we spiral in along streamlines that decrease the level of abstraction on each turn.

A strategy for software testing may also be viewed in the context of the spiral. Unit testing begins at the vertex of the spiral and concentrates on each unit of the software as implemented in source code. Testing progress is done by moving outward along the spiral to integration testing, where the focus is on the design and the construction of the software architecture. Talking another turn on outward on the spiral we encounter validation testing where requirements established as part of software requirements analysis are validated against the software that has been constructed. Finally we arrive at system testing, where the software and other system elements are tested as a whole.

**6.3 UNIT TESTING**

Unit testing focuses verification effort on the smallest unit of software design, the module. The unit testing we have is white box oriented and some modules the steps are conducted in parallel.

**6.3.1 WHITE BOX TESTING**

To follow the concept of white box testing we have tested each form. We have created independently to verify that Data flow is correct and all conditions are exercised to check their validity. All loops are executed on their boundaries.

This type of testing ensures that

* All independent paths have been exercised at least once
* All logical decisions have been exercised on their true and false sides
* All loops are executed at their boundaries and within their operational bounds
* All internal data structures have been exercised to assure their validity paths.

**6.3.2 CONDITIONAL TESTING**

In this part of the testing each of the conditions were tested to both true and false aspects. And all the resulting paths were tested so that each path that may generate on a particular condition is traced to uncover any possible errors.

**6.3.3 DATA FLOW TESTING**

This type of testing selects the path of the program according to the location of definition and use of variables. This kind of testing was used only when some local variable were declared. The definition-use chain method was used in this type of testing. These were particularly useful in nested statements.

**6.3.4 LOOP TESTING**

In this type of testing all the loops are tested to all the limits possible. The following exercise was adopted for all loops:

* All the loops were tested at their limits, just above them and just below them. All the loops were skipped at least once.
* For nested loops test the inner most loop first and then work outwards.
* For concatenated loops the values of dependent loops were set with the help of connected loop.
* Unstructured loops were resolved into nested loops or concatenated loops and tested as above.
* Each unit has been separately tested by the development team itself and all the input have been validated.

**6.4 TEST CASE**

A test case, in software engineering, is a set of conditions or variables under ~which a tester will determine whether an application, software system or one of its features is working as it was originally established for it to do. The mechanism for determining whether a software program or system has passed or failed such a test is known as a *test oracle*. In some settings, an oracle could be a requirement or use case, while in others it could be a heuristic. It may take many test cases to determine that a software program or system is considered sufficiently scrutinized to be released. Test cases are often referred to as *test* *scripts*, particularly when written - when they are usually collected intotest suites.

**Table6.1 Brief Test Case Description**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TEST** | **TEST** **DESCRIPTION** | **INPUT** | **EXPECTED RESULT** | **TEST RESULT** |
| 1 | Test the application when login page is opened. |  | System should display login screen with fields User ID and Password with Login button. | **Pass** |
| 2 | Test the application when the user clicks the login button after entering the correct details. | Email Id : user ID  Password : \*\*\*\*\*\*\* | The system should navigate to the map page. | **Pass** |
| 3 | Test the System when the user clicks the login button without entering the correct details. |  | The System should show error indicating wrong user ID and password. | **Pass** |
| 4 | Test the functionality of 'User ID' field | User mailID | The system should accept the input and should not throw any errors. | **Pass** |
| 5 | Test the functionality of 'Password' field | Password  \*\*\*\*\*\*\* | The system should accept the input and should not throw any errors. | **Pass** |
| 6 | Test the system when the user choose the nearest ambulance driver contact number. |  | The system must make a call to the corresponding ambulance driver. | **Pass** |
| 7 | Test the System when the driver log in to the application. |  | The system should navigate to the map page that shows user’s location. | **Pass** |
| 8 | Test the System when the driver accept the phone call. |  | The system should share the user’s location to the driver. | **Pass** |
| 9 | Test the System when the driver decline the phone call. |  | The system should inform the user to track another driver’s location. | **Pass** |

**Table6.2 Test Case Results**

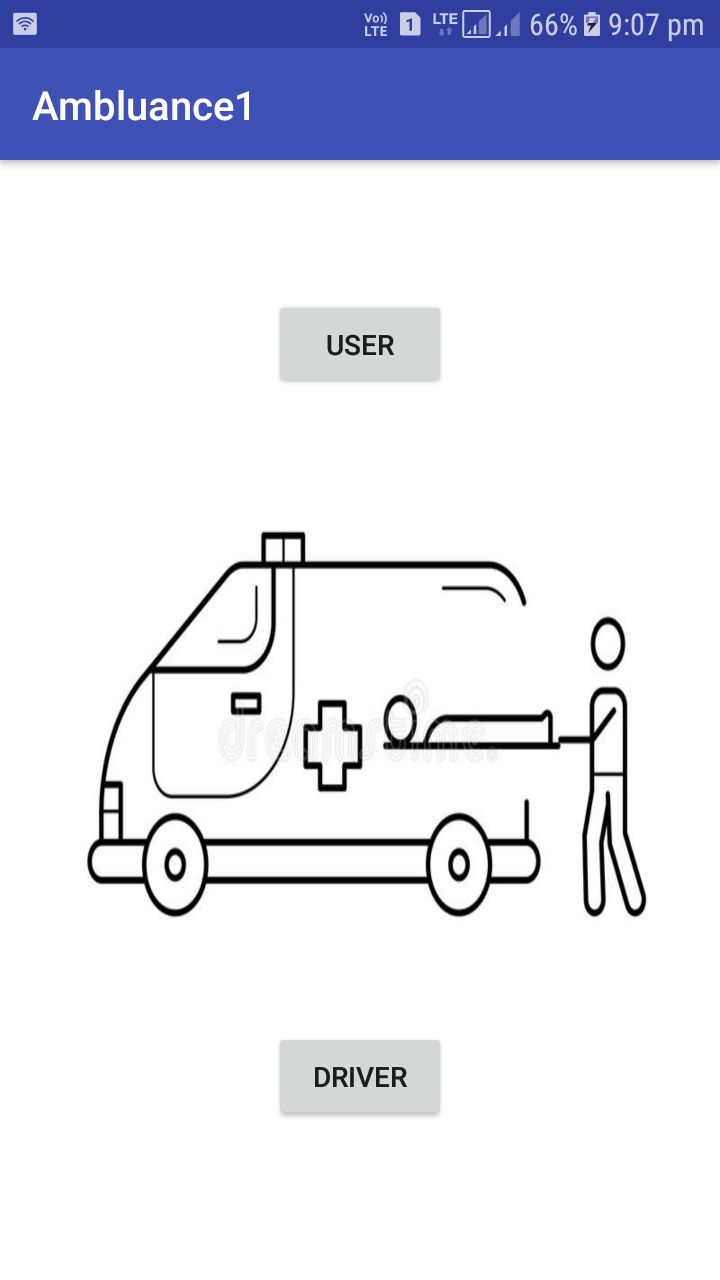
|  |  |  |  |
| --- | --- | --- | --- |
| **S NO.** | **TEST SCENARIO** | **EXPECTED RESULT** | **TEST RESULT** |
| 1 | Username is incorrect  Password is correct | Authentication failed. | Authentication failed. |
| 2 | Username is correct  Password is incorrect | Authentication failed. | Authentication failed. |
| 3 | Both username and password incorrect | Authentication failed. | Authentication failed. |
| 4 | Both username and password correct | Redirect to next page | Redirect to next page |
| 5 | Retrieve Data  (if any problem) | Fetch Data | Couldn’t get data |
| 6 | Retrieve Data | Fetch Data | Data received at the front end |

**CHAPTER 7**

**SCREENSHOTS**

**7.1 HOME PAGE**

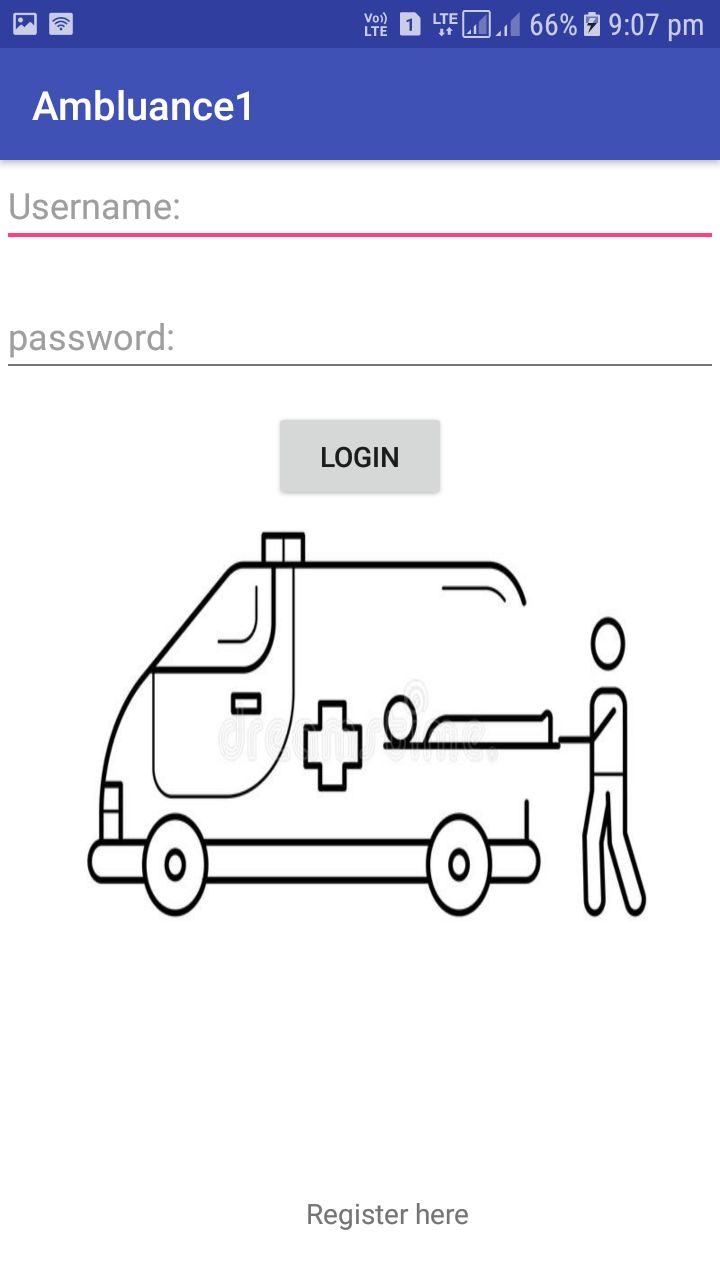
This page asks whether he/she is a user or an driver.



**Figure 7.1 Home Page**

**7.2 LOGIN PAGE**

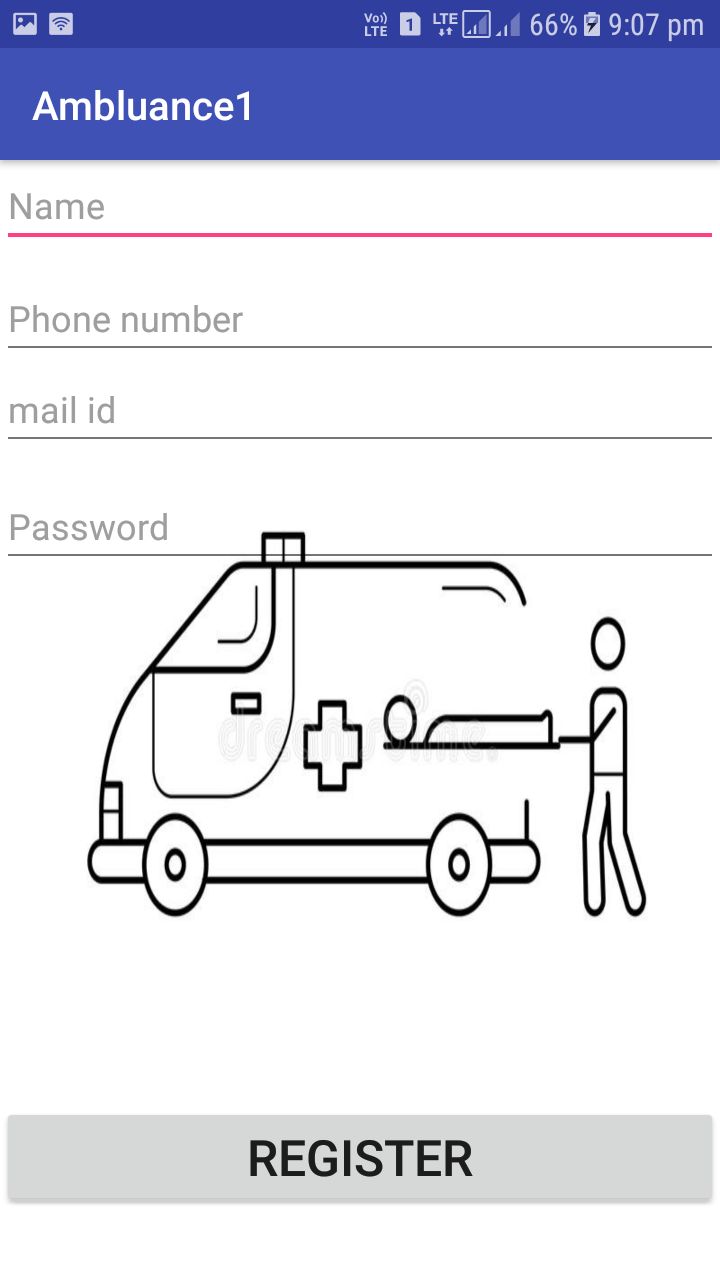
This page allows the user to login using mail id and password.

****

**Figure 7.2 Login Page**

**7.3 REGISTER PAGE**

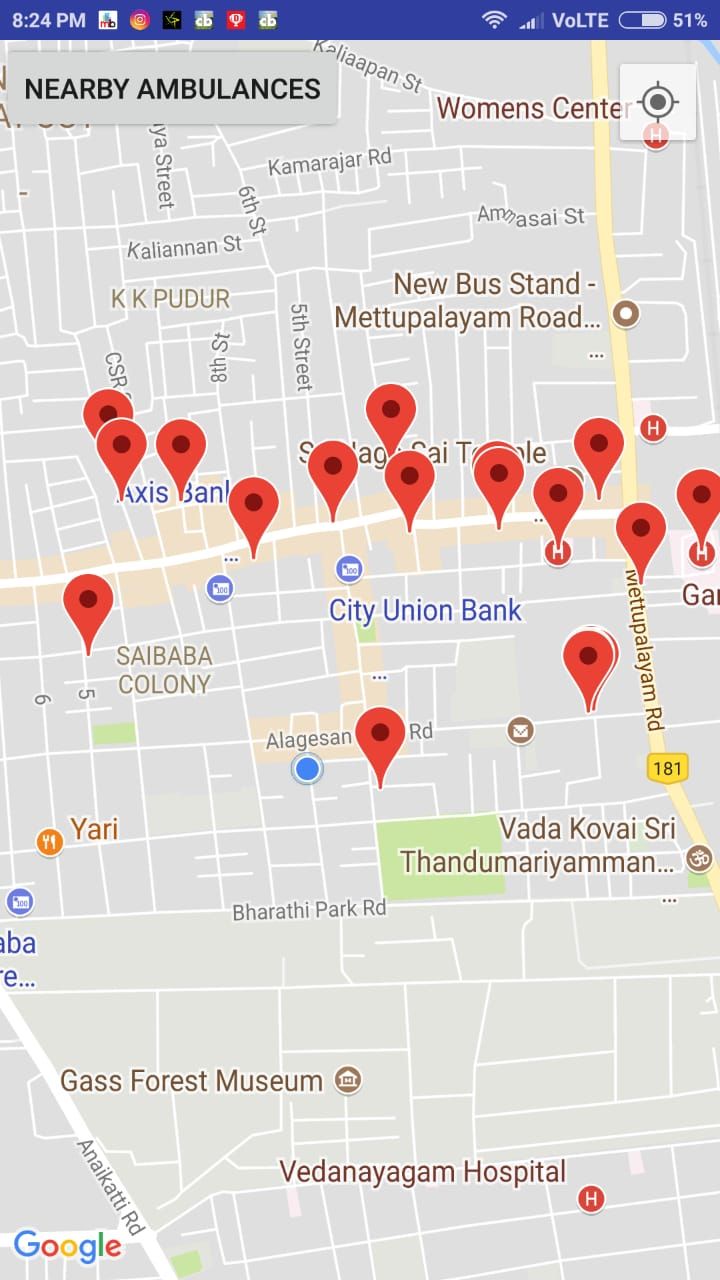
This page registers a new user by retrieving the name, phone number, mail id and password from the user.



**Figure 7.3 Register Page**

**7.4 MAP VIEW**

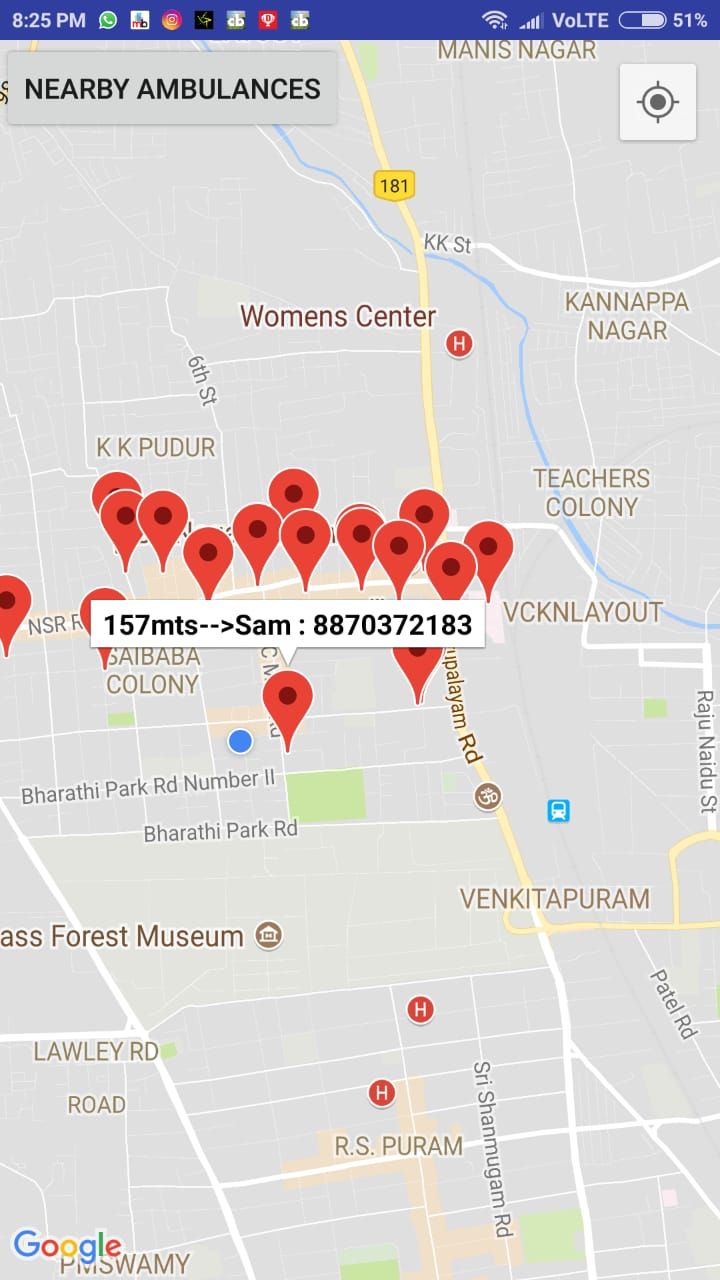
The user can view the current location in the map. The button at the top left corner is used to search the ambulances nearby.



**Figure 7.4 Map View**

**7.5 AMBULANCE DETAILS**

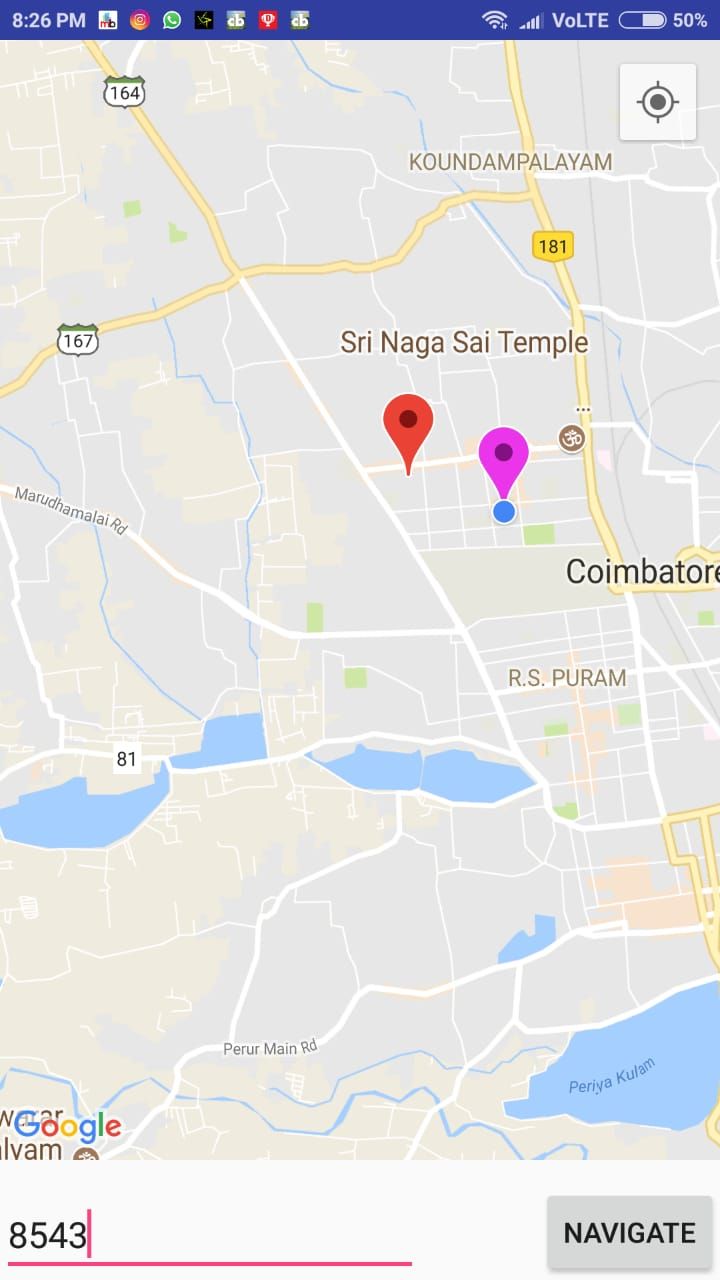
Clicking on a particular location of the ambulance, the user is provided with the distance from his location to that particular ambulance location along with driver name and his phone number.



**Figure 7.5 Ambulance Details**

**7.6 DRIVER MAP VIEW**

By getting the OTP from the user and filling in the text box the driver can view the users location and can navigate to that location.



**Figure 7.6 Driver Map View**

**CHAPTER 8**

**CONCLUSION**

**8.1 CONCLUSION**

When there is an emergency, first thing authorities would want to do is get the nearest team on spot at the earliest. Trying to call all ambulance drivers on road to know their locations is not the right way to go about it; it takes time. Thus with the help of proposed system**(Ambulance tracking System)**, you could find out the vehicles nearby the incident/accident location, just get the driver contact number and assign them the task. It is an easy portal to use and provide benefit to many people irrespective of their educational knowledge.

**8.2 FUTURE ENHANCEMENT**

* Initially it will track the nearest ambulance location alone. In future based on the success we will incorporate our system to locate nearby hospitals.
* So that we can locate nearby doctors for first aid.
* User can share any other location where help is needed apart from their own current location.

**CHAPTER 9**

**REFERENCES**

**WEBSITES**

* <https://www.tutorialspoint.com/android/android_studio.htm>
* <https://developer.android.com/guide/index.html>
* <https://developer.android.com/training/basics/firstapp/index.html>
* <https://www.simplifiedcoding.net/firebase-storage-example/>
* <https://www.lynda.com/Android-Studio-training-tutorials/6597-0.html>
* <https://in.udacity.com/course/new-android-fundamentals--ud851>
* <https://stackoverflow.com/questions/12664463/start-a-new-activity-from-non-activity-class>
* <https://stackoverflow.com/questions/29316662/how-do-i-locate-a-specific-location-in-google-map-view-in-android-studio>