

Konnected

SYNCHRONIZE ANDROID USERS

110141R

DODANGODA D.A.P.P

Contents

1.	INTRODUCTION	3
1.1.	Brief.....	3
1.2.	Introduction to functionalities.....	3
2.	LITERATURE REVIEW	4
3.	REQUIREMENT SPECIFICATION	8
3.1	Scope.....	8
3.2.	Definitions, Acronyms and Abbreviations.....	8
3.3.	Specific Requirements.....	8
3.3.1.	Functionalities.....	8
3.3.2.	Usability.....	9
3.3.3.	Performance.....	9
3.3.4.	Design Constraints	9
3.3.5.	Interfaces	9
4.	SYSTEM DESIGN	11
4.1.	Use Case View	11
4.2.	Logical View	12
4.3.	Architecturally significant Design Packages	12
4.4.	Process View.....	13
4.5.	Implementation View	14
4.6.	Quality and Performance	14
5.	SOFTWARE IMPLEMENTATION	15
7.	SYSTEM EVALUATION AND CONCLUSIONS	29

1. INTRODUCTION

1.1. Brief

Imagine a Pizza Restaurant which provides a delivery service. There can be several delivery boys working for the restaurant. The restaurant can get multiple requests for pizza for a single day. This will not be a problem as far as a single branch is considered. The delivery boy can mark a delivery as complete or in progress manually while visiting the branch to take pizza from there. So nobody else will try to do the same task.

But what will happen if there are multiple branches? If a user orders a pizza online, it will be shown to the delivery boys as a task to be completed. In this case two or more delivery boys can mistakenly try to deliver the same order without knowing that some other person is also trying to the same task. How can we prevent this? Can we get any help from technology?

“Konnected” is one such effort for avoiding and preventing real world synchronization conflictions. It uses Google Android technology for creating an Android application which can run on the android operating system.

1.2. Introduction to functionalities

The main function of this application is the synchronization of tasks as explained in the above scenario. Not just avoiding conflictions, but also sharing tasks is also taken into consideration.

To get a clearer visual of this, imagine an average home. There will be the family members, Father, Mother, and Children. Suppose they need to buy things for the house. For example, Sugar, Milk Powder, Rice, Books, Carpets etc. And need to pay some bills, like electricity bill, telephone bill, water bill etc. Likewise there can be many tasks which cannot be accomplished by one person, and also required by any of the members.

So with this application, any of the members can request some task to be accomplished. Seeing the update, any other person can accept the request and complete the task. Therefore, communication becomes easier because they don’t have to call and tell every other person to do it. And there is no risk of multiple people doing the same task. For example two people can pay the electricity bill through two different outlets without knowing that the other person also paid it. But with this application if one person is doing a task, he or she can indicate that he or she is doing it, so that no other person will try to do it at the same time.

So as explained with two scenarios, the basic idea is to maintain a shared resource, which is a to-do list in this case to which several allowed people can contribute at the same time. Key features of these functions are explained in detail in the following chapters of this report.

2. LITERATURE REVIEW

The Key Idea

In today's world, sharing information can be considered as the most requested and used service from software applications. This can be pointed out as the reason for spreading and growing the interest towards social media websites and applications. In specific, the highest interest is towards sharing dynamic information. Online chat applications can be considered as an example where dynamic information is shared. A very common feature that exists among almost all these sources is that these are applied only for general use and entertain. Yet this approach can be very useful and productive if used in the business world.

As explained in the practical scenarios, connecting and synchronizing of multiple android users can be very advantageous and productive in business societies as well as in general. Yet there are no applications so far which allow a fully customizable feature of connecting users. Of course there are several applications which use this principal for their own tasks.

Many companies have embarked on initiatives that enable more demand information sharing between retailers and their upstream suppliers. While the literature on such initiatives in the business press is proliferating, it is not clear how one can quantify the benefits of these initiatives and how one can identify the drivers of the magnitudes of these benefits. Using analytical models, this paper aims at addressing these questions for a simple two-level supply chain with non stationary end demands.^[1]

Target Platform

In today's world, one can say that everything depends on time. Even the earnings and profits of a business system depend on how efficient the system is. Therefore people value time over every other aspect. Mobile phones and similar mobile devices were invented as a solution to save money. Therefore now it's the task of software developers to make those high performance hardware devices even better by creating better and efficient software to run on them.

One of the most successful mobile operating systems is the Google Android operating system which is a free to use open source OS. Therefore I selected android OS as the platform for this application.

Developing software with the android SDK was a new challenge. Here is a brief introduction to android SDK and app development.

Android SDK^[3]

Android powers hundreds of millions of mobile devices in more than 190 countries around the world. It's the largest installed base of any mobile platform and growing fast—every day another million users power up their Android devices for the first time and start looking for apps, games, and other digital content.

Android gives you a world-class platform for creating apps and games for Android users everywhere, as well as an open marketplace for distributing to them instantly.

Android SDK is a software development kit that enables developers to create applications for the Android platform. The Android SDK includes sample projects with source code, development tools, an emulator, and required libraries to build Android applications. Applications are written using the Java programming language and run on Dalvik, a custom virtual machine designed for embedded use which runs on top of a Linux kernel.^[4]

Eclipse ADT

I have chosen the eclipse ADT as the IDE for the software project. The Eclipse ADT bundle includes everything you need to begin developing apps:

- Eclipse + ADT plugin
- Android SDK Tools
- Android Platform-tools
- A version of the Android platform
- A version of the Android system image for the emulator

Robotium Solo

I have used the Robotium solo 5.2.1 to test the android application I developed. It is a tool which uses automated testing to test for the functionality of different functions as well as activities.

Robotium is an Android test automation framework that has full support for native and hybrid applications. Robotium makes it easy to write powerful and robust automatic black-box UI tests for Android applications. With the support of Robotium, test case developers can write function, system and user acceptance test scenarios, spanning multiple Android activities.^[5]

Robotium provides the following benefits:

- Test Android apps, both native and hybrid.
- Requires minimal knowledge of the application under test.
- The framework handles multiple Android activities automatically.
- Minimal time needed to write solid test cases.
- Readability of test cases is greatly improved, compared to standard instrumentation tests.
- Test cases are more robust due to the run-time binding to UI components.
- Fast test case execution.
- Integrates smoothly with Maven, Gradle or Ant to run tests as part of continuous integration.

Approach

The approach for developing this application is pretty much similar to the incremental model. Incremental model in software engineering is a one which combines the elements of waterfall model which are then applied in an iterative manner. It basically delivers a series of releases called increments which provide progressively more functionality for the client as each increment is delivered ^[2].

The suggested increments can be listed as

- Basic set of android Activities
- To-Do Activity with hardcoded To-Do Tasks
- Data Input Activity which gives the option to add a new activity
- Map Activity with the features related to geographical location
- Notification system for the tasks (Time and Location)
- Additional settings

Related Developments and Products available

Following is a list of extremely popular applications which connect multiple users. Nevertheless it can be noted that it is not their primary task. Connecting users is done in a way such that only a small requirement is fulfilled.

- 1) Facebook : A social media application. Allows users to create and organize user groups. There are options to get notifications and share information. But there is no clearly defined way to create small-medium scale tasks which contain unique details related to the task. Facebook has several features related to geographical location, but no option to affiliate them with tasks.
- 2) WhatsApp: It can share information such as texts, images and audio. Can be considered as a chat application. It does not allow multiple users to get into one account (Max-2).

- 3) Drive: Allows sharing static data among multiple users. Cannot share dynamic resources and get related notifications

Listed above are three of the most popular applications which allow information sharing. Yet it is to be noted that currently none of these or other applications provide the expected functionality of the “Konnected” application.

3. REQUIREMENT SPECIFICATION

This section describes the Requirement Specification which was launched at the initial phase of the project. Please note that there can be a few differences from the following requirements in the final product which was released recently.

3.1 Scope

This is an android application which allows several smartphone users to join a personal network and synchronize details about their geographical position and some other related features.

3.2. Definitions, Acronyms and Abbreviations

Android	– Android™ is an operating system which is used by many mobile devices
ADT	– Android™ Development Tools
SDK	– Software Development Kit
3G	– 3rd Generation
IDE	– Integrated Development Environment
API	–Application Programming Interface
Maps	–Maps is an Application Programming Interface provided by Google™ for adding google map components to mobile applications

3.3. Specific Requirements

3.3.1. Functionalities

3.3.1.1. Log in and Sign up

Users can log in to the application by giving valid a user name and a password, or can create an account if it does not exist. Users can join to a specific network by giving the required credentials.

3.3.1.2. Shared To-Do List

Users can add a To-Do task by giving the required parameters (Title, Description, Priority, Location on the Map, etc). It will be updated on every other user's device. Then another user can mark the task as "acquired" and literally start doing the intended task. After completion he or she can mark the task as "Accomplished". If anything goes wrong, it can be marked as "Failed" so that another user can acquire the task.

3.3.1.3. Location based notifications

Users can get location based notifications about the to-do tasks when he or she is closer to a location which is marked on any of the to-do tasks. The notification message may include the following details

- Task Name and description
- Location details and distance from current location
- The user who initiated the task
- current progress of the task

3.3.1.4. Shortest Path

Shortest path and other related details such as shortest path distance and time to travel can be calculated and/or displayed on a map.

This can be related to shortest path to a to-do task or the shortest path to some other member of the group.

3.3.2. Usability

Any smart phone user with android operating system can easily operate the application. Additionally, the application will be designed in such a way that unnecessary interaction by the users is not expected. An internet connection is a must for this application to perform correctly. A functionality to retrieve data from the central database and update a local database can be added later.

3.3.3. Performance

Continuous internet connectivity is required as the users are synced through the internet. We can expect latencies in cases where the internet connection is not speed enough. An Android device has many methods to connect internet such as Wi-Fi, GPRS, and 3G. This application must support all these connecting methods in order to function in every case possible.

3.3.4. Design Constraints

Android SDK

Android SDK must be used for developing android applications. There is no other alternative for this. Android developers provide Android SDK which is free to use.

Eclipse IDE

Eclipse has a plug-in for android development which makes easy to design user interfaces in collaboration with Java SE. Other IDEs like NetBeans do not offer a wide range of plug-ins for Android.

3.3.5. Interfaces

User Interfaces

A user can log into the app via the login screen. Then the user will be forwarded to the main screen where the user can join a team. Also, there will be a user friendly icon set to perform the functionalities provided by the application.

Shortest Path- This will request the name of a specific member and will display the shortest path on the google map when given

Observe- This provides functionality to one of the location based notifications. This will request the name of a specific member, a location, and also an action which can be selected from a dropdown menu.

E.g. – Notify me when <UserName> Leaves/Reaches <location>

Checklist This will navigate to another screen where the checklist will display the corresponding location, status, and the contributing user(s)

Hardware Interfaces

Android device which supports internet connectivity and SD storage card is sufficient.

Software Interfaces

Application may be developed for the Android platform version 4.0. Since Android has backward compatibility new versions of Android will be able to execute this product.

Communications Interfaces

HTTP protocol is used for feed downloads. For establishing internet connection Wi-Fi, 3G or GPRS must be used.

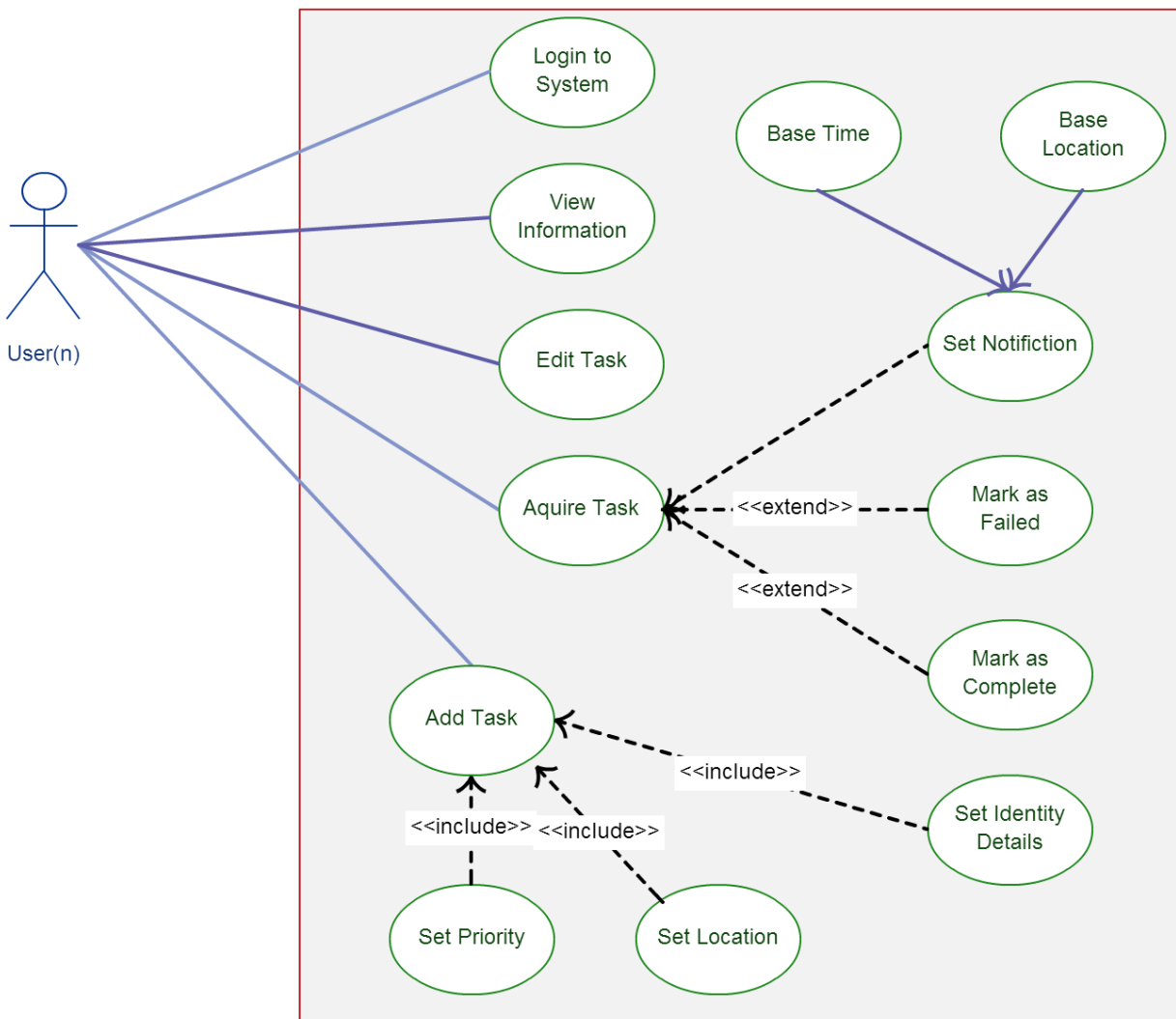
.....

P.T.O.

4. SYSTEM DESIGN

This section describes the requirements and objectives that have some significant impact on the architecture of the application.

4.1. Use Case View

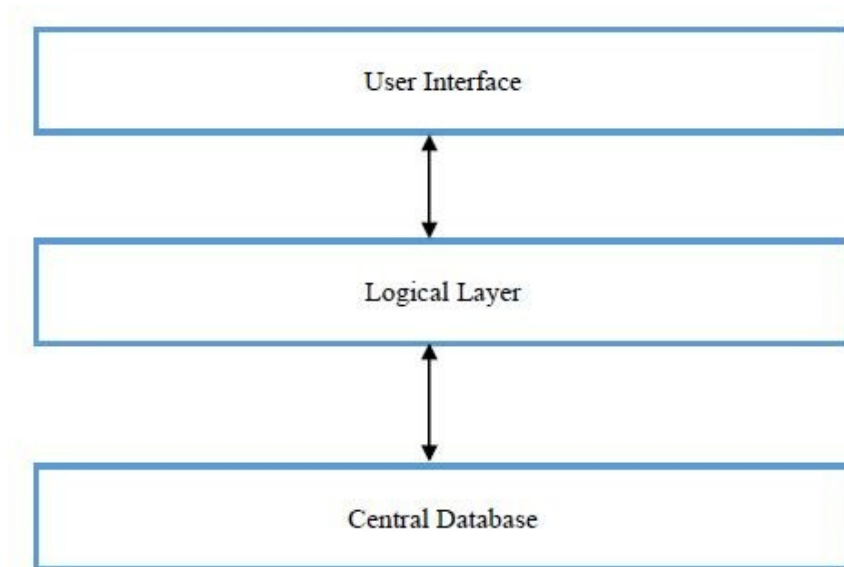


The users can log into an account and once they have, they can choose any from several available options. The main 3 functions as per described in the above diagram are, Add Task, Acquire Task and update ToDo list. Other general functions such as leaving the account and set visibility are also possible. The main 3 use cases are complex use cases which include and extend several other use cases.

For Example, when a user sets a notification, he or she may request it based upon time or location.

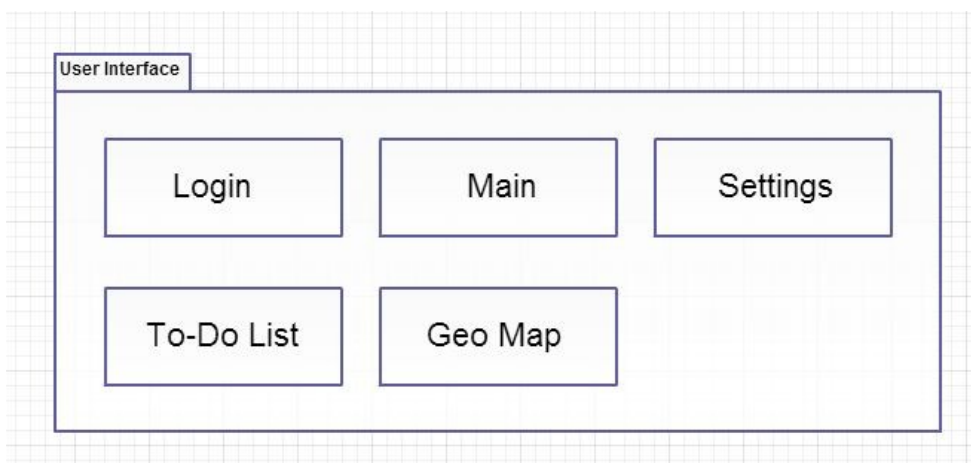
4.2. Logical View

This application consists of 3 main layers. They are User Interface, Logical Layer and Central Database. Top layer is the User Interface layer which is used by the user to interact with the system. Middle layer is the logical layer. The logical layer manages communication between users and the system. The database layer stores data about To-Do tasks, Notifications, the user accounts, and user behaviors.

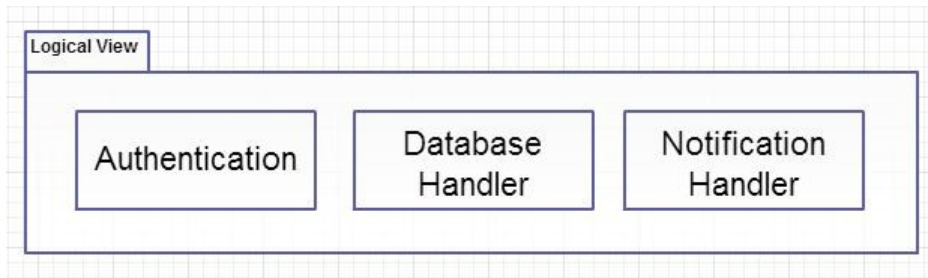


4.3. Architecturally significant Design Packages

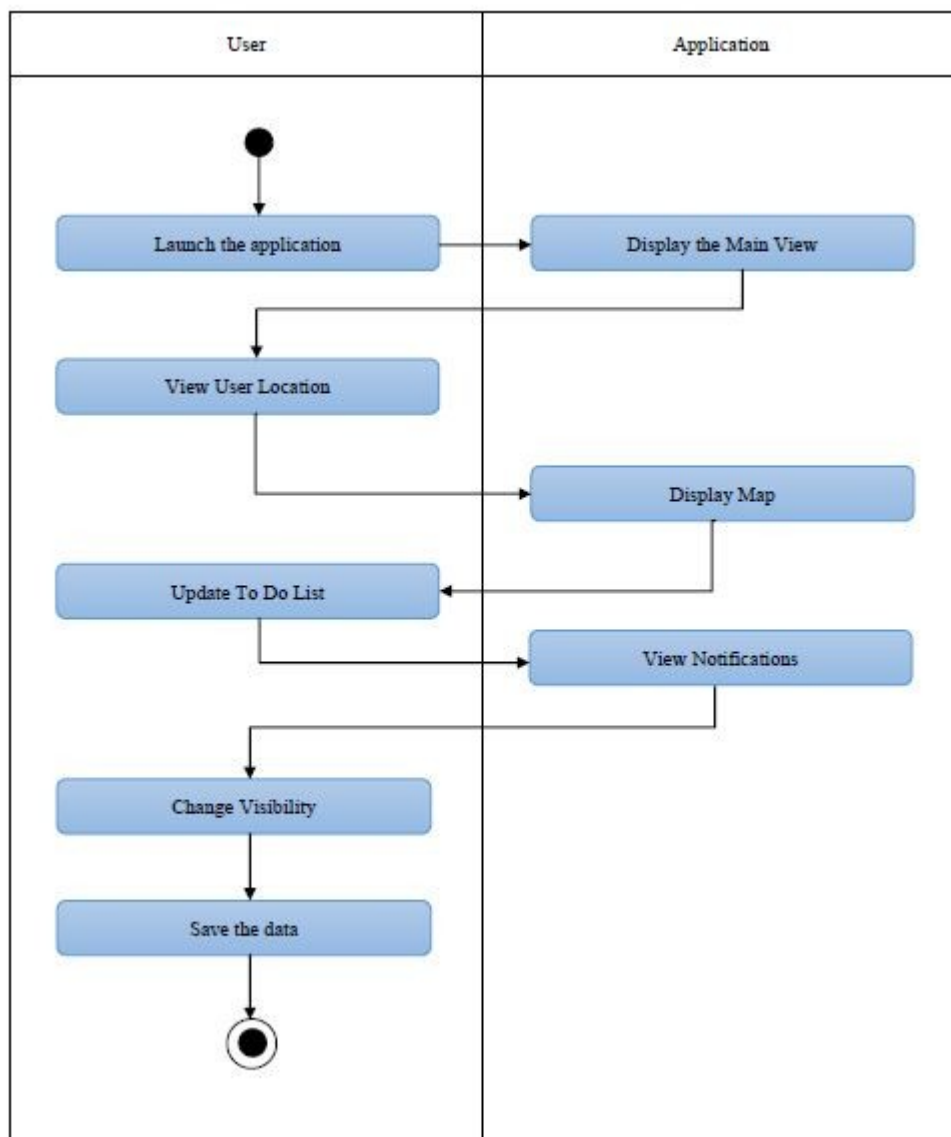
UI Layer



Logical Layer



4.4. Process View



4.5. Implementation View

Overall structure of the implementation model, the decomposition of the software into layers and subsystems in the implementation model of this system is same as the logic view of the system.

UI Layer

UI layer comprises of classes extended from the Activity class. Those classes provide User Interface. Also it displays the outputs of logic layer.

Logic Layer

Main responsibility of this layer is communication between users and managing their behaviors and the To Do list.

4.6. Quality and Performance

Overall quality of the system is improved because of the layered architecture. Capabilities that are improved by the software architecture are

- Performance.
- Increased abstraction level.
- Reusability.
- Extendibility.

Performance mainly depends on the bandwidth and the signal strength of the internet connection. Memory will also be used for rendering maps. Since there is a notification system, the application will need to be running in the background. So as a whole, the app will need comparatively high performance hardware support.

5. SOFTWARE IMPLEMENTATION

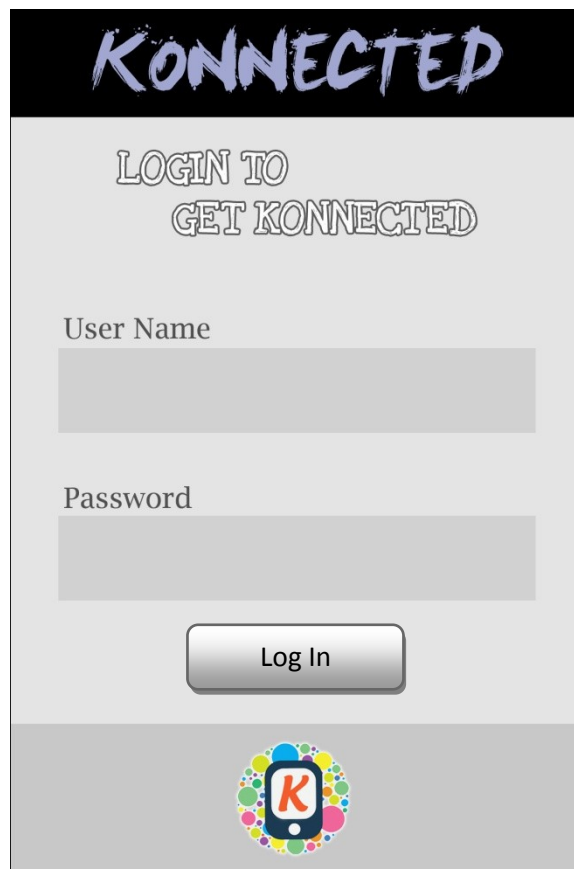
As explained earlier the software was basically coded with Java SE. This section explains the implementation process of the software step by step. The application software was developed according to the RUP model. But in this section details related to requirement analysis, designing and testing are discarded though all these were done in parallel.

Phase 1- Logging in

The initial focus was on the overall structure of the application. It was decided to design separate xml activity for each important function of the software.

As a first step the login activity was designed using xml and the java code was written to move from this activity to the main activity when the required credentials are given.

In the main activity 4 image buttons were created to move to the main functions of the application



The following code was written to test the login function without using database access. Therefore hard coded credentials were used.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.Login1);
    getActionBar().hide();
    Button signin = (Button) findViewById(R.id.button1);    //button to start main activity
    signin.setOnClickListener(new OnClickListener() {        //on click actions of sign in button

        @Override
        public void onClick(View v) {
            Intent intent = new Intent(LoginActivity.this,Main.class);
            EditText userName=(EditText) findViewById(R.id.Text1); //to read the given username
            EditText password=(EditText) findViewById(R.id.Text2); //to read the given password

            if(userName.getText().toString()==null){    //check for null username
                return;
            }

            if(userName.getText().toString().equals("Pubudu")||userName.getText().
                toString().equals("Dodan")){
                return;    //only in app coded usernames are accepted. SShould be extended to
            }

            if(!password.getText().toString().equals("pass")){
                return;    //only in app coded password is accepted. SShould be extended to s
            }

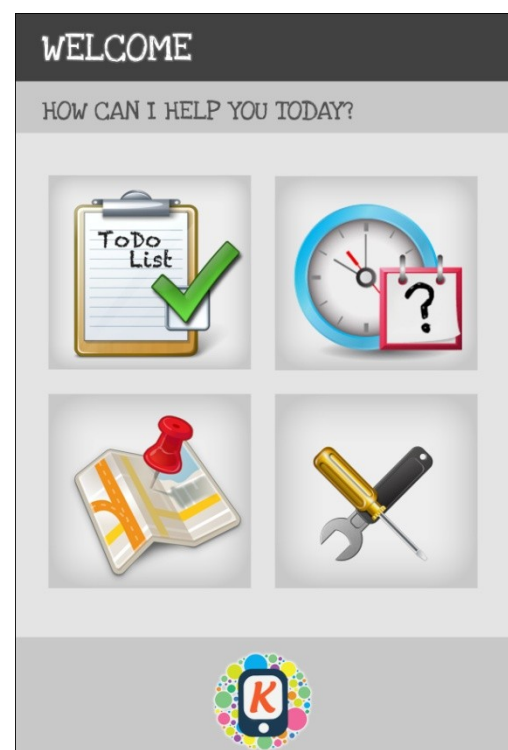
            intent.putExtra("username", userName.getText().toString()); //pass the username to the
            startActivity(intent); //start main activity

        }
    });
}
```

After a successful session of this task, it was decided to implement the rough structure of all the activities.

The four main activities forwarded from the main activity are as follows,

- ToDo Activity
- Map Activity
- Notification Activity
- Setting Activity



Phase 2 – Todo List

The main function of the application was initiated at this stage, the shared to-do List.

This was mostly based on the java code instead of xml designs with contrast to phase1. A class was implemented with the necessary attributes to hold all the information related to an item in the to-do list.

```
public class ToDoItem {                                //To do items.Creates an item
                                                        //of this to be added to the todo item list

    private int entityNo;                                //attributes of the to do item
    private int position;
    private String title;
    private String description;
    private int priority;
    private String latitude;
    private String longitude;
    private int status;
```

The basic constructors, getter and setter methods were included though they are not shown here.

However an entry of the to-do list was not just an object of this class. An adapter class was used to hold the necessary information and display the entry.

Inside the adapter class an inner class with name “holder” was used to hold required info. The only intention of creating this class is the ease of access.

Every object created from the ToDoItem class was added to an ArrayList and the data was retrieved inside the adapter class for creating the entry.

```
@Override
public View getView(int index, View convertView, ViewGroup parent) {

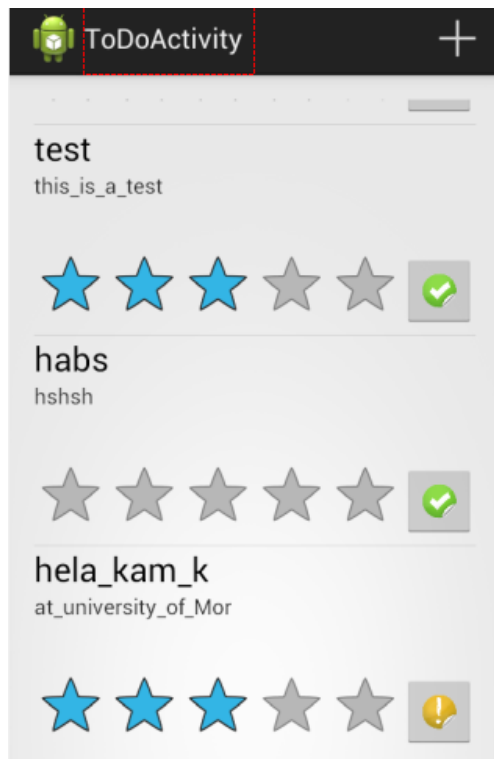
    View vi = convertView;
    ViewHolder holder;

    if (convertView == null) {
        vi = inflater.inflate(R.layout.draw_item, null);
        holder = new ViewHolder();
        holder.title = (TextView) vi.findViewById(R.id.viewTitle); // title
        holder.description = (TextView) vi.findViewById(R.id.description); // description
        holder.menu = (ImageView) vi.findViewById(R.id.image);
        holder.priority=(RatingBar)vi.findViewById(R.id.ratingBar1);
        holder.statusBtn=(ImageButton)vi.findViewById(R.id.image);
        vi.setTag(holder);
    } else {
        holder = (ViewHolder) vi.getTag();
    }

    ToDoItem item = data.get(index);
    final int entNo=index;

    // Setting all values in listview
    holder.title.setText(item.getTitle());
    holder.description.setText(item.getDescription());
    holder.priority.setRating(item.getPriority());
```

After creating an object with the above code, what will appear on the android screen is some object similar to what is shown below.



The item contains a title, a description about the item, priority of the task and the status of the task.

Status of the task is the most important feature of this task item. When a user taps on the status button, the image as well as the database should be updated. That is the current status should be switched between the three options, Inprogress(acquired), Accomplished and Failed(Not started). The three icons used to show the status as explained are shown below



Figure 1: Acquired Task



Figure 2: Accomplished Task



























Figure 3: Failed Task

Phase 3 – Database connection

As the next step, the database connection was setup with the application. A MySQL database was created with a table named todo items. This table contains all the information which should be included in a todo task item.

The importance of the database should be highlighted. The main function of this software application is to connect several android users into one system. So to do that a shared data center is a must. This is achieved through the centralized database.

The following screenshot shows the structure of the table used for this section.

<input type="checkbox"/>	1	EntryNumber	int(11)	No	None	 Change	 Drop	 Primary	▼ More
<input type="checkbox"/>	2	Position	int(11)	No	None	 Change	 Drop	 Primary	▼ More
<input type="checkbox"/>	3	Title	varchar(20) latin1_swedish_ci	No	None	 Change	 Drop	 Primary	▼ More
<input type="checkbox"/>	4	Description	varchar(60) latin1_swedish_ci	No	None	 Change	 Drop	 Primary	▼ More
<input type="checkbox"/>	5	Priority	int(11)	No	None	 Change	 Drop	 Primary	▼ More
<input type="checkbox"/>	6	Latitude	varchar(10) latin1_swedish_ci	No	None	 Change	 Drop	 Primary	▼ More
<input type="checkbox"/>	7	Longitude	varchar(10) latin1_swedish_ci	No	None	 Change	 Drop	 Primary	▼ More
<input type="checkbox"/>	8	Status	int(11)	No	None	 Change	 Drop	 Primary	▼ More

Communication of information with the database was done with php. The php code used for retrieving data from the table is shown below

```
<?php
$con=mysql_connect("localhost","root","");

// Check connection

if ($con == null) {
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

if (!mysql_select_db("Konnected")) {
    echo "Unable to select Konnected: " . mysql_error();
    exit;
}

$result = mysql_query("SELECT * FROM ToDoItems");
while($row = mysql_fetch_assoc($result))
{
    $output[]=$row;
}

print(json_encode($output));
mysql_close($con);
?>
```

For this task there was no need to give any input to the php file. But later, it will be noted that inputs should be given, when inserting and updating data to the table, to identify intended column(s).

To read data from the table, the following method should be called. It should be highlighted that what is returned by calling this method is an Array of JSON Objects. A single JSON object will consist of data in a single row. Therefore the array will contain all the data retrieved from the table.

```
public JSONArray getItemData(){

    String url="http://192.168.43.46/Konnect/getToDoItem.php?"; //access php,
    HttpEntity httpentity=null;
    JSONArray dataArray=null;

    try{
        HttpClient httpclient= new DefaultHttpClient();
        HttpGet httpGet=new HttpGet(url);
        HttpResponse httpResponse = httpclient.execute(httpGet);
        StatusLine line = httpResponse.getStatusLine();// get the response sta
        if(line.getStatusCode() == 200){//if status is valid
            httpentity=httpResponse.getEntity();//the the response entity
            if(httpentity != null){
                try{
                    String entityResponse=EntityUtils.toString(httpentity);//c
                    dataArray= new JSONArray(entityResponse);//put the String
                }
                catch(JSONException e){
                    e.printStackTrace();
                }
                catch(IOException e){
                    e.printStackTrace();
                }
            }
        }else{
        }
    }
    catch(ClientProtocolException e){
        e.printStackTrace();
    }
    catch(IOException e){
        e.printStackTrace();
    }
    return dataArray; //return the list of to do items read as an array of J
}
```

The link to the php file on the server can be directly given to read data. In this case The IP address of the server machine is 192.168.43.46.

The returned JSON array must be processed to read the required data retrieved from the table. The following method was called to process the read JSON object.

```
public void organize(JSONArray jarray){
    JSONObject json=null;
    try{
        int numOfObj=jarray.length();           //number of to do items in the database

        for(int i=0;i<numOfObj;i++){

            json=jarray.getJSONObject(i);    //reads each item, object by object
            int entityNo=Integer.parseInt(json.getString("EntryNumber"));           //rea
            int position=Integer.parseInt(json.getString("Position"));               //rea
            String title=json.getString("Title");                                   //rea
            String description=json.getString("Description");                       //rea
            int priority=Integer.parseInt(json.getString("Priority"));               //rea
            String latitude=json.getString("Latitude");                             //rea
            String longitude=json.getString("Longitude");                           //rea
            int status=Integer.parseInt(json.getString("Status"));                   //rea

            ToDoItem item=new ToDoItem(entityNo,position,title,description,priority,
                latitude,longitude,status);           //creates a new todo item
            items.add(item);                           //add to do item to the l

        }
        ToDoAdapter adapter=new ToDoAdapter(this, items);    //wrap the list in an ada

        ListView listview=(ListView)findViewById(R.id.listView1);

        listview.setAdapter(adapter);           //add each adapter to a l
        lastEntry=jarray.getJSONObject(numOfObj-1).getInt("EntryNumber");           //read th
                                                    //of the
                                                    //when ad
        System.out.println("Last entry : "+ lastEntry);           //print t

    }
    catch(Exception e){
        System.out.println(e);    //print exception to the log cat
    }
}
```

However these methods being directly called can lower the performance of the application. Therefore an inner class is used for this task.

This so called class should be inherited from the AsyncTask class

```

private class GetAllItems extends AsyncTask<GetItem, Void, JSONArray>{//this class connect
    @Override
    protected void onPreExecute() {
        // TODO Auto-generated method stub
        super.onPreExecute();
    }

    protected JSONArray doInBackground(GetItem... param){// connect the internet and get th
        return param[0].getItemData();
    }

    @Override
    protected void onPostExecute(JSONArray result) {
        organize(result);
    }
}

```

Shown above is the code of the inner class explained earlier.

Writing to the database

A similar approach is taken to write data to the database. The php code used for this task is shown below

```

<?php
$con=mysql_connect("localhost","root","");
$ent=$_GET['entityNo'];
$pos=$_GET['position'];
$tit=$_GET['title'];
$des=$_GET['description'];
$pri=$_GET['priority'];
$lat=$_GET['latitude'];
$lon=$_GET['longitude'];
$sta=$_GET['status'];
// Check connection
if ($con == null) {
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}
if (!mysql_select_db("Konected")) {
    echo "Unable to select Konected: " . mysql_error();
    exit;
}
$result = mysql_query("INSERT INTO `todoitems`(`EntryNumber`,
    `Position`, `Title`, `Description`, `Priority`, `Latitude`,
    `Longitude`, `Status`) VALUES ($pos,$ent,\"$tit\", \"$des\",
    $pri,\"$lat\", \"$lon\", $sta)");
#mysql_fetch_assoc($result);
mysql_close($con);
?>

```


It is to be noted here that there is an option to give in inputs to this php code. A sample url that can be used to input data to this is shown below

```
String url="http://192.168.43.46/Konnect/setToDoItem.php?" +
    "&entityNo="+this.entityNo+
    "&position="+this.position+
    "&title="+this.title+
    "&description="+this.description+
    "&priority="+this.priority+
    "&latitude="+this.latitude+
    "&longitude="+this.longitude+
    "&status="+this.status; //access php, database with th
```

Details related to AsyncTask inherited class and stuff is common to this as well. Since it is quite similar, it is not explained here in detail

Updating the Database

The database should be updated when a user changes the status of a todo task as explained earlier in this section. A similar approach is used to update the database. The php code used for that is shown below.

```
<?php
$con=mysql_connect("localhost","root","");

$ent=$_GET['entityNo'];
$sta=$_GET['status'];

// Check connection

if ($con == null) {
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

if (!mysql_select_db("Konnect")) {
    echo "Unable to select Konnect: " . mysql_error();
    exit;
}

$result = mysql_query("UPDATE `todoitems` SET `Status`=$sta
    WHERE EntryNumber=$ent");
#mysql_fetch_assoc($result);
echo $ent;
echo $sta;

mysql_close($con);
?>
```

Phase 4 – Configuring Google Map

The next important task done was the configuration of the map activities. A location is added to each todo item as another attribute. The initial intention was to get location based notifications from the todo-Tasks added. Before everything else, the functionality to get user location and intended location should be implemented.

The following code segment was used for this task.

```
Fragment frag = getFragmentManager().findFragmentById(R.id.map);

MapFragment mf = (MapFragment) frag.getFragmentManager()
    .findFragmentById(R.id.map);
mMap = mf.getMap();

mMap.setMyLocationEnabled(true);

mMap.setOnMapClickListener(new GoogleMap.OnMapClickListener() {

    @Override
    public void onMapClick(LatLng point) {

        if (marker != null) {
            marker.remove();
        }
        MarkerOptions mo = new MarkerOptions().position(point);
        marker = mMap.addMarker(mo);
        lat=marker.getPosition().latitude+"";
        lat=lat.substring(0, 4);
        lon=marker.getPosition().longitude + "";
        lon=lon.substring(0,4);
        ((TextView)findViewById(R.id.lat)).setText(lat);
        ((TextView)findViewById(R.id.lon)).setText(lon);

        mMap.moveCamera(CameraUpdateFactory.newLatLng(point));
    }
});
```

This code will successfully generate the map and the ability to click on a location and mark it. However, the map will still be in a zoomed out status as shown below.



To avoid this problem and to zoom to the current location or to the middle of Sri Lankan map, the following code segment was used.

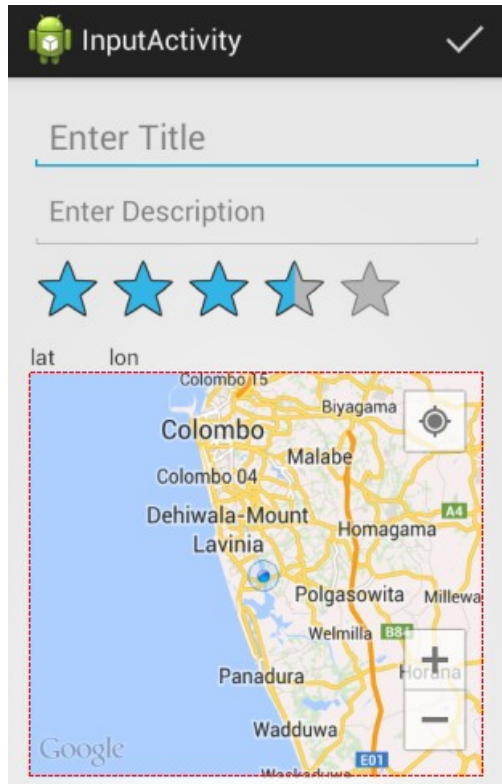
```
try{
    Criteria criteria = new Criteria();
    LocationManager locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
    String provider = locationManager.getBestProvider(criteria, false);
    Location location = locationManager.getLastKnownLocation(provider);
    LatLng coordinate = new LatLng(location.getLatitude(), location.getLongitude());
    CameraUpdate yourLocation = CameraUpdateFactory.newLatLngZoom(coordinate, 10);
    mMap.animateCamera(yourLocation);
}
catch(Exception e)
{
    LatLng coordinate = new LatLng(7.38, 80.77);
    CameraUpdate yourLocation = CameraUpdateFactory.newLatLngZoom(coordinate, 7);
    mMap.animateCamera(yourLocation);
}
```

For this to happen as intended the permissions should be given from the android manifest.

Phase 5 – Adding Map related functions to the ToDo List

After the correct configuration of the map, the required functions should be added to the application. Out of them the main requirement is to get the intended location to the input of creating a todo item for the list.

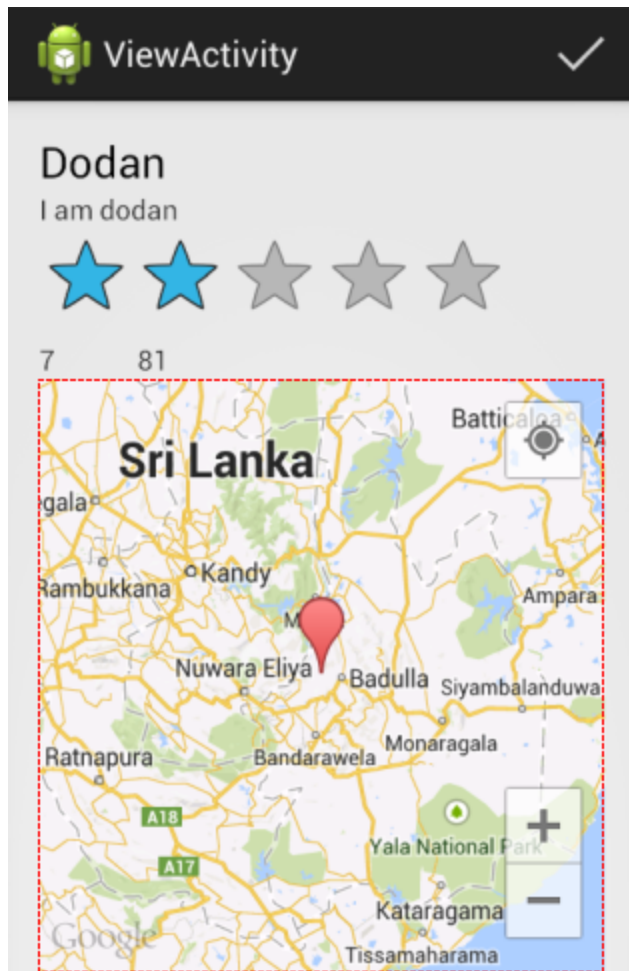
The following screenshot shows how the application interface looks when this functionality is added.



Phase 6 – View ToDo Item

Another functionality was added later to view the added tasks. When the title is clicked a new activity window appears with the added details, including the latitudes longitudes and the priority ratings. This window is not editable at the moment

A screen shot of this window is shown below



Phase 7 – Location Based Notifications

Location based notifications were added to the to-do tasks in the final phase of the application.

When a user reaches a location of the marked to-do items, he or she will receive an android notification with a reminder to this task including the priority of the task

```
@Override
protected void onHandleIntent(Intent arg0) {

    while(!stop){
        LocationResult locationResult =
new LocationResult(){
            @Override
            public void gotLocation(Location location){
                //Got the location!
            }
        };
        MyLocation myLocation = new MyLocation();
        myLocation.getLocation(this,
locationResult);
        try {
            Thread.sleep(30000);
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

}
```

7. SYSTEM EVALUATION AND CONCLUSIONS

In parallel to the implementation of the application, Testing was carried out as a major activity. Types of testing used can be summarized as below.

- 1) Manual Testing
- 2) Automated Unit Testing
- 3) Beta Testing

Here, manual testing refers to the tests performed directly via the code. Most people find it difficult to debug and test android applications since it does not give us a console which pops up error messages as well as system print streams.

However the LOGCAT which is provided for this task can be used in a similar way to do manual testing. When the code meets up a `System.out.println();` type code segment, the logcat shows that part as a printstream to the log.(usually in green color)

However automated testing cannot be omitted in an application related to android systems. As explained in the literature review, a tool was used to test this application with unit testing.

Robotium requires the developer to write test cases for the application.

Listed below are 5 of the test cases used to test this application.

- **Test 1**

```
public void test1(){
    solo.clickOnButton("Sign In");
    solo.assertCurrentActivity("Test Activity", Main.class);
    solo.goBack();
}
```

- **Test 2**

```
public void test2(){
    solo.clickOnButton("Sign In");
    solo.assertCurrentActivity("Test Activity", Main.class);
    solo.clickOnImageButton(0);
    solo.assertCurrentActivity("To Do Activity Appear", ToDoActivity.class);
    solo.goBack();
    solo.goBack();
}
```

- Test 3

```
public void test3(){
    solo.clickOnButton("Sign In");
    solo.assertCurrentActivity("Test Activity", Main.class);
    solo.clickOnImageButton(3);
    solo.assertCurrentActivity("Setting Activity Appear", SettingsActivity.class);
    solo.goBack();
    solo.goBack();
}
```

- Test 4

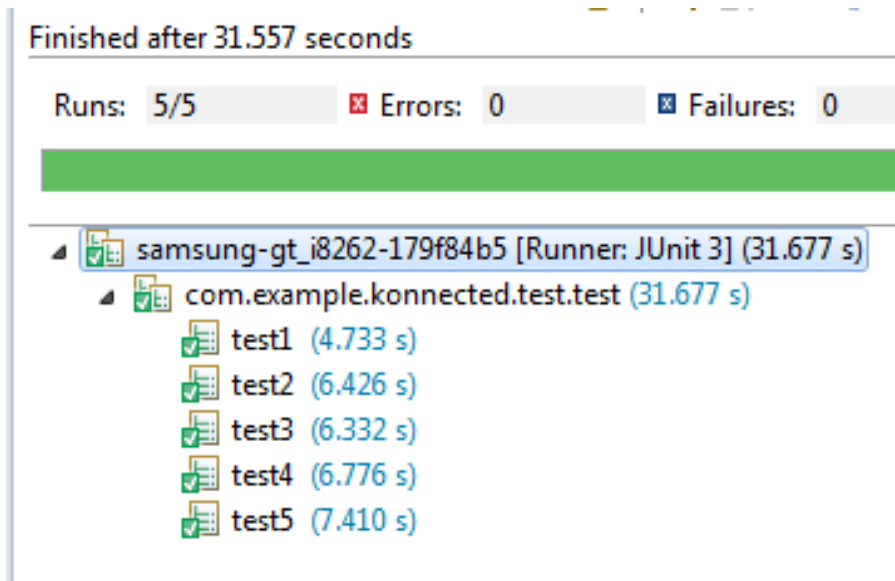
```
public void test4(){
    solo.clickOnButton("Sign In");
    solo.assertCurrentActivity("Test Activity", Main.class);
    solo.clickOnImageButton(2);
    solo.assertCurrentActivity("Map Activity Appear", MapActivity.class);
    solo.goBack();
    solo.goBack();
}
```

- Test 5

```
public void test5(){
    solo.clickOnButton("Sign In");
    solo.assertCurrentActivity("Test Activity", Main.class);
    solo.clickOnImageButton(0);
    solo.assertCurrentActivity("To Do Activity Appear", ToDoActivity.class);
    solo.clickOnActionBarItem(0);
    solo.assertCurrentActivity("Input", InputActivity.class);
    solo.goBack();
    solo.goBack();
    solo.goBack();
}
```

Automated test Result

The final result of the automated testing carried out by Robotium Solo is shown below



Beta testing was carried out by several volunteer android users and one of the major problems raised was the screen compatibility. That is due to different resolutions present in these devices, the interface showed different problems. Therefore the xml files were changed again to avoid this issue.

Conclusions

The Final decision would be to consider this application as a success as almost all the expected functionalities were accomplished by the end of the time frame.

Few key things that were noted and would be taken into consideration are as follows.

- The application must be improved to support concurrency
- Due to the less target customer set, several other features should be added to attract more users
- The database should be hosted on a real server instead of WAMP and then beta testing and alpha testing should be carried out to improve performance

References

[1] Lee, Hau L., Kut C. So, and Christopher S. Tang. "The value of information sharing in a two-level supply chain." *Management science* 46.5 (2000): 626-643.

[2] Rahul Tilloo. (2013) Incremental Model In Software Engineering[online]. Available: <http://www.technotrice.com/incremental-model-in-software-engineering>

[3] Android, the world's most popular mobile platform. [Online] Available: <http://www.developer.android.com/about/start.html>

[4] Android SDK.[online]. Available: http://www.webopedia.com/TERM/A/Android_SDK.html

[5] Robotium, User scenario testing for Android. [Online] Available: <https://code.google.com/p/robotium/>