**Chapter 1**

**INTRODUCTION**

Nowadays, usage of mobile has become a vital part of day-to-day activities of people. We can refer the current time as the era of Smartphone’s. Suppressing all other traditional communication purpose, Smartphone’s are now at the peak of popularity in their usage of accessing the internet which includes mail access, social networking, mobile shopping and mobile banking. Given these features the problem arises when the phone is left at someplace and we want to retrieve some data from the Smartphone. Remote accessing of mobile becomes necessary in such cases. To deal with this problem, we thought of developing an application that permits remote accessing. It includes getting the incoming call numbers, incoming messages, accessing call logs, changing phone’s profile setting, retrieving the IMEI number of the cell phone, retrieving the sim serial number as well as retrieving the contacts.

This application is designed for android users and is useful in the following scenarios.

1) You reach your office and realize that you have forgotten your phone at your home. Need to urgently call a client whose phone number is stored in mobile. Not a problem, with this app on your phone you can easily access and retrieve the address book of your phone remotely by sending a simple SMS command to your phone number  from your colleagues/friends any model cell phone. This app will take the effort to find your required number and SMS it back to you on your sent mobile number.

2) You are in the office and have misplaced your phone somewhere nearby. Can’t find it even by calling on your number as your phone is on silent mode. Frustrated…, not anymore, you can now remotely turn your phone on RINGER mode by sending an SMS command from friends phone. Easily find your phone and be relaxed! Similarly you can even turn your phone on SILENT mode and also you can easily get details of the last calls and messages received on your mobile phone without having it.

**Chapter 2**

**LITERATURE REVIEW**

The mobile phones have evolved from simple communication devices to responsible units which can handle mission critical applications. The usage and security aspects of mobile devices are important. Smart phones are being used in almost all parts of your day to day life by making tasks easier and affordable. Some applications have concentrated on accessing the smart phones remotely.

* 1. Norton Anti theft plug-in enables the users to locate and lock the mobile through any web browser. It also provides facility to take picture through the camera of the Smartphone and send to their online storage [5].

**Features:**

* **Locate:** Tracks and maps the location of your device when anyone uses it to go online so you will know if it is lost or stolen.
* **Remote Lock:** Securely locks your device to prevent strangers from using it or seeing your information.
* **Lost Notification:** Displays a customizable message to anyone who finds your missing device so you can make arrangements to get it back.
* **Web Control:** Works securely over the Internet, so you can easily track, locate, and lock your devices anytime, anywhere.
* **Sneak Peek:** Snaps photos of anyone or anything in front of your device then saves the images to the Norton Anti-Theft Web site to help you get back your device. (Webcam devices only).

2.2 McAfee WaveSecure also provides the similar functionality and it enables the users to erase their personal data in smartphone through online. McAfee WaveSecure secures your mobile device and personal data in the event of loss or theft by allowing you to locate/track your device, helps prevent device misuse, and remotely locks your device and wipes your data. It enables you to back up and restore your data from the device or via an easy-to-use web portal[6].

**FEATURES**

* Lock & Wipe  
  - Remotely lock your device from the web or via SMS from another device  
  - Display a message on your locked device  
  - Wipe your personal data
* Backup & Restore  
  - Back up your data to the cloud (even after device loss or theft)  
  - Restore data to a new device  
  - Access your data online from anywhere
* Locate & Track  
  - Locate your lost device and plot its locations on a map  
  - Sound an alarm to help retrieve your missing device  
  - Track the current SIM and calls made to help recover your device

These applications need computer to access smartphone and moreover these are concerned with mobile theft only.

The proposed application offers more features than existing one. Our app can not only turn on the ringer of your phone remotely but will also send you the phone’s current GPS location by SMS. It provides the core functionalities of the applications mentioned above also. This model uses SMS as the communication channel. The permission issues on Android also need to be considered while implementing the remote access.

**Chapter 3**

**PROBLEM DEFINITION AND OBJECTIVES**

The existing applications require computer to remotely access and secure Smartphone’s our focus is on developing an application that allows remote accessing of Smartphone through a normal mobile.

Having this app installed on your phone magically allows you to access your mobile remotely. You can access & retrieve a lot of data from your mobile phone even if you don’t have it with you.

The data which could be fetched includes:

* Contacts (Fetching contact number from your address book)
* Call Logs (Checking missed calls/received calls/dialed numbers)
* IMEI Number
* Phone Profile (Changing profile mode to ringer or silent mode)
* SIM Number (Retrieving SIM Number)
* SMS Logs (Checking SMS’s received on your phone)
* SIMOP
* Location of the device

**Chapter 4**

**REQUIREMENTS**

Following are the requirements to be considered while designing this application

4.1 Technical Requirements

* Hardware Requirements
* Cell phone as SMSC
* Android Smart Phone.
* A standard computer with atleast
* 512 MB RAM.
* 2 MB HDD free space for installing our application.
* 100 MB HDD free space for installing necessary software.

* Software Requirements
* JDK(Java Development Kit) version greater than 1.5
* Android Development Tools (ADT)
* SQLITE database
  1. Economical Requirements

Smartphone usage has become more popular these days because of its advanced features that are very helpful and are also affordable.

* 1. Feasibility

The hardware and software requirements are minimum and cost effective and also the cost of Smartphone’s is reducing day by day. Hence the application is feasible considering all the aspects required to build it.

4.4 Tools

Our primary development platform consisted of the following software and hardware configuration:

4.4.1 Android SDK.

The Android SDK provides a set of integrated development and testing tools including core Android libraries, a built in emulator, debugger, logger, and device drivers to allow running applications on an Android phone. We used the Google API add-on libraries for implementing some of features in LBR such as geo-coding, mapping, etc.

4.4.2 Eclipse Plug-ins.

The Eclipse IDE v3.4.2 was used with the Android Development (ADT) and Subversion plug-ins. The ADT plug-in allows developing and running apps from within the Eclipse IDE and provides access to various features of the Android SDK. The Subversion plug-in was used for working with the SVN repository from within Eclipse IDE.

4.4.3 Android Phone.

Any android phone with 2.3.6 Android Gingerbread OS.

**Chapter 5**

**Design**

5.1 Process to input secret key

The application has been designed in such a way that the user is asked to input a secret key or pin which will be used to establish the connection with the remote mobile. Once the pin is accepted the application will run in the background and it will wait for incoming messages.

iii Ask the user to input secret key

IS Is key provided?

no

EE Exit

Exit

yes

Run the app in background and wait for the new incoming new message

Fig 5.1 Process to input secret key

5.1.1 Algorithm for the process to input the secret key

STEP 1: Start

STEP 2: Ask the user to input secret key

STEP 3: If key is provided, go to step 4,Else exit

STEP 4: Run the app in the background and wait for new incoming messages

5.2 Activity in background process

Now you are ready to access your mobile phone remotely. So whenever you forget your mobile phone on your home, Send a SMS to your mobile number with the Security PIN to access your Android Device. Example your Security pin is 1234 and Mobile number 9769688838. Then SMS PIN 1234 to 9769688838.

Upon reading incoming message the app will verify if the key provided in the message is same to the one set during installation. If it is same the application will send the response with all possible process list. The list includes options for retrieving the following details.

* Contacts (Fetching contact number from your address book)
* Call Logs (Checking missed calls/received calls/dialed numbers)
* IMEI Number
* Phone Profile (Changing profile mode to ringer or silent mode)
* SIM Number (Retrieving SIM Number)
* SMS Logs (Checking SMS’s received on your phone)
* SIMOP
* Location of the device

Once the user chooses the option and sends it to remote mobile the app will check if it is valid. If it is valid, it will process the request and send the response as a message.

RE Read incoming message

Check for authentication using the unique key

IS Is user authenticated??

no

SI End the request

yes

S Send response to message with all possible process list

R Read the option provided by user and check if valid

i Is valid option?

no

yes

P Process the request

R Return the process result as message

Fig 5.2.1 Flowchart of the background process

5.2.1 Algorithm for the background process

STEP 1: Start

STEP 2: Read incoming message

STEP 3: Check for authentication using the unique key

STEP 4: If user is authenticated go to step 5, otherwise end the request

STEP 5: Send response to message with all possible process list

STEP 6: Read the option provided by user and check if valid

STEP 7: If the option is valid go to step 8, otherwise go to step 10

STEP 8: Process the request

STEP 9: Return the process result

STEP 10: Stop

5.3 Steps to retrieve the data stored in smartphone

Enter the remote connection command to establish connection with the android device.

5.3.1 Steps to retrieve contacts

Step 1: After entering the connection command wait for the application to return the list of options.

Step 2: Select the option corresponding to retrieve the contacts and send it as a message to the remote mobile.

Step 3: wait for the app to return the process result.

5.3.2 Steps to retrieve call logs

Step 1: After entering the connection command wait for the application to return the list of options.

Step 2: Select the option corresponding to retrieve the call logs and send it as a message to the remote mobile.

Step 3: wait for the app to return the process result.

5.3.3 Steps to retrieve IMEI number

Step 1: After entering the connection command wait for the application to return the list of options.

Step 2: Select the option corresponding to retrieve the IMEI number and send it as a message to the remote mobile.

Step 3: wait for the app to return the process result.

5.3.4 Steps to change the phone profile

Step 1: After entering the connection command wait for the application to return the list of options.

Step 2: Select the option corresponding to silent/ringer and send it as a message to the remote mobile.

Step 3: wait for the app to return the process result.

5.3.5 Steps to retrieve SIM number

Step 1: After entering the connection command wait for the application to return the list of options.

Step 2: Select the option corresponding to retrieve the SIM number and send it as a message to the remote mobile.

Step 3: wait for the app to return the process result.

5.3.6 Steps to retrieve SMS

Step 1: After entering the connection command wait for the application to return the list of options.

Step 2: Select the option corresponding to retrieve the SMS’s and send it as a message to the remote mobile.

Step 3: wait for the app to return the process result.

5.3.7 Steps to retrieve SIM operator

Step 1: After entering the connection command wait for the application to return the list of options.

Step 2: Select the option corresponding to retrieve the SIM operator and send it as a message to the remote mobile.

Step 3: wait for the app to return the process result.

5.3.8 Steps to retrieve the location of mobile

Step 1: After entering the connection command wait for the application to return the list of options.

Step 2: Select the option corresponding to retrieve the location and send it as a message to the remote mobile.

Step 3: wait for the app to return the process result.

After we are done with the process, we need to send a STOP command to terminate the connection.

5.4 Use case diagram showing the interaction between user and smartphone

smartphone

user

**Chapter 6**

**IMPLEMENTATION**

The proposed model for smartphone remote access has been implemented in the android 2.3.3 platform. Mobile users have to type the command from messaging interface of their mobile. The following commands are implemented.

**Table 6.1: List of commands in the prototype implementation**

|  |  |
| --- | --- |
| **COMMAND** | **DESCRIPTION** |
| LOCATION | TO get the current cell phone location |
| SIMNO | To get the sim serial number |
| SIMOP | To get the sim operator |
| IMEI | To get the cell IMEI number |
| SILENT | To set ringer mode to silent |
| RINGER | TO set silent mode to ringer |
| LMESG n | To get last nth received message |
| CALLS | To get the recent call details |
| FIND Name | To get the contact details |
| STOP | To terminate the connection |

To aceess a smartphone the user should input the remote connection command and send it to source mobile.

For example ,if the user is having pin “1234” then he has to type PIN<space>1234.This will give the protection.

### 6.1 Activity In Android

•An activity presents a visual user interface for one focused endeavor the  user can undertake.

•An application might consist of just one activity or many.

•Each one is implemented as a subclass of the Activity base class.

An [Activity](http://developer.android.com/reference/android/app/Activity.html) is an application component that provides a screen with which users can interact in order to do something, such as dial the phone, take a photo, send an email, or view a map. Each activity is given a window in which to draw its user interface.

An application usually consists of multiple activities that are loosely bound to each other. Typically, one activity in an application is specified as the "main" activity, which is presented to the user when launching the application for the first time. Each activity can then start another activity in order to perform different actions. Each time a new activity starts, the previous activity is stopped, but the system preserves the activity in a stack (the "back stack"). When a new activity starts, it is pushed onto the back stack and takes user focus.

6.1.1 Declaring Activity In Manifest

<manifest ... >

<application ... >

<activity android:name=".ExampleActivity" />

...

</application ...>

</manifest >

6.1.2  Using Intent Filters  In Activity Tag

<activity android:name=".ExampleActivity" android:icon="@drawable/app\_icon">

<intent-filter>

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

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

</intent-filter>

</activity>

The [<action>](http://developer.android.com/guide/topics/manifest/action-element.html) element specifies that this is the "main" entry point to the application.

The [<category>](http://developer.android.com/guide/topics/manifest/category-element.html) element specifies that this activity should be listed in the system's application launcher (to allow users to launch this activity).

### 6.1.3 Starting An Activity:

#### You can start another activity by calling [startActivity()](http://developer.android.com/reference/android/app/Activity.html#startActivity%28android.content.Intent%29), passing it an [Intent](http://developer.android.com/reference/android/content/Intent.html) that describes the activity you want to start. The intent specifies either the exact activity you want to start or describes the type of action you want to perform

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

startActivity(intent);

### 6.1.4 Implementing the lifecycle callbacks:

### When an activity transitions into and out of the different states described above, it is notified through various callback methods. All of the callback methods are hooks that you can override to do appropriate work when the state of your activity changes. The following skeleton activity includes each of the fundamental lifecycle methods:

public class ExampleActivity extends Activity {

@Override

public void [onCreate](http://developer.android.com/reference/android/app/Activity.html#onCreate%28android.os.Bundle%29)(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

// The activity is being created.

}

@Override

protected void [onStart()](http://developer.android.com/reference/android/app/Activity.html#onStart%28%29) {

super.onStart();

// The activity is about to become visible.

}

@Override

protected void [onResume()](http://developer.android.com/reference/android/app/Activity.html#onResume%28%29) {

super.onResume();

// The activity has become visible (it is now "resumed").

}

@Override

protected void [onPause()](http://developer.android.com/reference/android/app/Activity.html#onPause%28%29) {

super.onPause();

// Another activity is taking focus (this activity is about to be "paused").

}

@Override

protected void [onStop()](http://developer.android.com/reference/android/app/Activity.html#onStop%28%29) {

super.onStop();

// The activity is no longer visible (it is now "stopped")

}

@Override

protected void [onDestroy()](http://developer.android.com/reference/android/app/Activity.html#onDestroy%28%29) {

super.onDestroy();

// The activity is about to be destroyed.

}

}

An activity is a single, focused thing that the user can do. Almost all activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI with setContentView(View)

6.1.5 An Activity represents an UI with which a user can Interact

Each activity in an application goes through its own lifecycle. Once and only once when an activity is created, is the onCreate() function executed. If the activity exits, the  
onDestroy() function is executed. In between, various events can lead to the activity  
being in multiple different states, as illustrated in Figure.

O onCreate()

A Activity launched

User navigates to the activity

onRestart()

onStart()

onResume()

A App process killed

A Activity running

r

User returns to the activity

Another activity comes into the foreground

onPause()

User naviga tes to activity The activity is no longer visible

onStop()

Apps with higher priorty need memory

The activity is finishing or being destroyed by the system

onDestroy()

A Activity shutdown

Fig 6.1.5.1 Lifecycle of an activity

As seen here, various common actions by the user can cause the activity to be paused,  
killed,or even launch multiple versions of the application.

* **onCreate() :** Called when the activity is first created. This is first method called when an Activity  starts. This is where you should do all of your normal static set up: create views, bind data to lists, etc. This method also provides you with a Bundle containing the activity's previously frozen state, if there was one. Generally in this method we write setContentView()  method and inflate the XML layout.
* **onStart() :** Called when the activity is becoming visible to the user.
* **onResume() :** Called when the activity will start interacting with the user. At this point your activity is at the top of the activity stack,with the user input going to it.
* **onPause() :** Called when when you start a New Activity , the previous Activity goes in Pause state and pushed in stack.  When we come back on previous  Activity , Previous Activty gets Poped from Stackand onResume() method of previous Activity gets called.
* **onStop() :** Called when the activity is no longer visible to the user, because another activity has been resumed and is covering this one. This may happen either because a new activity is being started, an existing one is being brought in front of this one, or this one is being destroyed.
* **onDestroy() :** called when the  your activity is going to be destroyed.

6.2 Receiving Text Message

The first thing we're going to do is to give the app permission to receive SMS messages. Just add the uses permission android.permission.RECEIVE\_SMS using the manifest editor or the xml tag

<uses-permission android:name="android.permission.RECEIVE\_SMS">

directly into the manifest file.

Now that we can receive messages, we need to register an **Intent Broadcast Receiver** that will be called whenever a message arrives.When messages are received, the onCreate() method will be invoked.The message is contained and attached to the Intent object (intent - the second parameter in the onReceive() method) via a Bundle object. The messages are stored in an Object array in the PDU format. To extract each message, you use the static createFromPdu() method from the SmsMessage class.The SMS message is then displayed using the Toast class:

In our example we'll use the following class:

class SMSReceiver extends BroadcastReceiver{

@Override

public void onReceiver(Context context, Intent intent){

//do something with the message received

}

}

To register a broadcast receiver, in the application section of manifest file add

<receiver android:name=".SMSReceiver">

<intent-filter>

<action android:name="android.provider.Telephony.SMS\_RECEIVED"/>

</intent-filter>

</receiver>

The receiver has an intent filter tag, meaning that only that type of intent will be passed to the intent receiver. Android will make sure the receiver code runs even if your app is not running.

Now let's see how to get the content of the the received message.

The SMS messages in Android use the PDU format (Protocol Description Unit) meaning that the content (sender, body, etc) is available in text mode albeit encoded. But fear not, the API can easily decode PDU data.

The message received by the broadcast receiver is available in the extras of the intent under the tag "pdus".

The SMS protocol allows for 160 characters long messages only. If the message is longer than that,several PDU objects will be created.So we need to assemble a message broken down into several PDUs.

6.3 Sending Text Message

To send an SMS message, you use the SmsManager class. Unlike other classes, you do not directly instantiate this class; instead you will call the getDefault() static method to obtain an SmsManager object.The sendTextMessage() method sends the SMS message with a PendingIntent. The PendingIntent object is used to identify a target to invoke at a later time. For example, after sending the message, you can use a PendingIntent object to display another activity.

sms.sendTextMessage(msg\_data[0], **null**,message, **null**, **null**);

##### The following Parameters are considered while sending the message.

|  |  |
| --- | --- |
| destinationAddress | the address to send the message to |
| scAddress | is the service center address or null to use the current default SMSC |
| text | the body of the message to send |
| sentIntent | if not NULL this PendingIntent is broadcast when the message is successfully sent, or failed. The result code will be Activity.RESULT\_OK for success, or one of these errors: RESULT\_ERROR\_GENERIC\_FAILURE RESULT\_ERROR\_RADIO\_OFF RESULT\_ERROR\_NULL\_PDU For RESULT\_ERROR\_GENERIC\_FAILURE the sentIntent may include the extra "errorCode" containing a radio technology specific value, generally only useful for troubleshooting. The per-application based SMS control checks sentIntent. If sentIntent is NULL the caller will be checked against all unknown applications, which cause smaller number of SMS to be sent in checking period. |
| deliveryIntent | if not NULL this PendingIntent is broadcast when the message is delivered to the recipient. The raw pdu of the status report is in the extended data ("pdu"). |

##### Throws

|  |  |
| --- | --- |
| [IllegalArgumentException](http://developer.android.com/reference/java/lang/IllegalArgumentException.html) | if destinationAddress or text are empty |

Send a multi-part text based SMS. The callee should have already divided the message into correctly sized parts by calling divideMessage.

##### Parameters

|  |  |
| --- | --- |
| destinationAddress | the address to send the message to |
| scAddress | is the service center address or null to use the current default SMSC |
| parts | an ArrayList of strings that, in order, comprise the original message |
| sentIntent | if not null, an ArrayList of PendingIntents (one for each message part) that is broadcast when the corresponding message part has been sent. The result code will be Activity.RESULT\_OK for success, or one of these errors: RESULT\_ERROR\_GENERIC\_FAILURE RESULT\_ERROR\_RADIO\_OFF RESULT\_ERROR\_NULL\_PDU For RESULT\_ERROR\_GENERIC\_FAILURE each sentIntent may include the extra "errorCode" containing a radio technology specific value, generally only useful for troubleshooting. The per-application based SMS control checks sentIntent. If sentIntent is NULL the caller will be checked against all unknownapplications, which cause smaller number of SMS to be sent in checking period. |
| deliveryIntent | if not null, an ArrayList of PendingIntents (one for each message part) that is broadcast when the corresponding message part has been delivered to the recipient. The raw pdu of the status report is in the extended data ("pdu"). |

Throws

|  |  |
| --- | --- |
| [IllegalArgumentException](http://developer.android.com/reference/java/lang/IllegalArgumentException.html) | if destinationAddress or text are empty |

6.4 Create a background service

The [IntentService](https://developer.android.com/reference/android/app/IntentService.html) class provides a straightforward structure for running an operation on a single background thread. This allows it to handle long-running operations without affecting your user interface's responsiveness. Also, an [IntentService](https://developer.android.com/reference/android/app/IntentService.html) isn't affected by most user interface lifecycle events, so it continues to run in circumstances that would shut down an [AsyncTask](https://developer.android.com/reference/android/os/AsyncTask.html).

An [IntentService](https://developer.android.com/reference/android/app/IntentService.html) has a few limitations:

1) It can't interact directly with your user interface. To put its results in the UI, you have to send them to an [Activity](https://developer.android.com/reference/android/app/Activity.html).

2) Work requests run sequentially. If an operation is running in an [IntentService](https://developer.android.com/reference/android/app/IntentService.html), and you send it another request, the request waits until the first operation is finished.

3) An operation running on an [IntentService](https://developer.android.com/reference/android/app/IntentService.html) can't be interrupted.

6.4.1 Define the IntentService in the Manifest

An [IntentService](https://developer.android.com/reference/android/app/IntentService.html) also needs an entry in your application manifest. Provide this entry as a [<service>](https://developer.android.com/guide/topics/manifest/service-element.html) element that's a child of the [<application>](https://developer.android.com/guide/topics/manifest/application-element.html) element:

**<service android:name=*"Operator"*></service>**

Now that you have the basic [IntentService](https://developer.android.com/reference/android/app/IntentService.html) class, you can send work requests to it with [Intent](https://developer.android.com/reference/android/content/Intent.html) objects.

# 6.4.2 Sending Work Requests to the Background Service

We have seen how to create an [IntentService](https://developer.android.com/reference/android/app/IntentService.html) class. The explaination below shows how to trigger the [IntentService](https://developer.android.com/reference/android/app/IntentService.html) to run an operation by sending it an [Intent](https://developer.android.com/reference/android/content/Intent.html). This [Intent](https://developer.android.com/reference/android/content/Intent.html) can contain optionally contain data for the [IntentService](https://developer.android.com/reference/android/app/IntentService.html) to process. You can send an [Intent](https://developer.android.com/reference/android/content/Intent.html) to an [IntentService](https://developer.android.com/reference/android/app/IntentService.html) from any point in an [Activity](https://developer.android.com/reference/android/app/Activity.html) or [Fragment](https://developer.android.com/reference/android/app/Fragment.html)

## 6.4.3 Create and Send a Work Request to an IntentService

## To create a work request and send it to an [IntentService](https://developer.android.com/reference/android/app/IntentService.html), create an explicit [Intent](https://developer.android.com/reference/android/content/Intent.html), add work request data to it, and send it to [IntentService](https://developer.android.com/reference/android/app/IntentService.html) by calling [startService()](https://developer.android.com/reference/android/content/Context.html#startService%28android.content.Intent%29).

Once you call [startService()](https://developer.android.com/reference/android/content/Context.html#startService%28android.content.Intent%29), the [IntentService](https://developer.android.com/reference/android/app/IntentService.html) does the work defined in its [onHandleIntent()](https://developer.android.com/reference/android/app/IntentService.html#onHandleIntent%28android.content.Intent%29) method, and then stops itself.

# 6.4.4 Reporting Work Status

# To send the status of a work request in an [IntentService](https://developer.android.com/reference/android/app/IntentService.html) to other components, first create an [Intent](https://developer.android.com/reference/android/content/Intent.html) that contains the status in its extended data. As an option, you can add an action and data URI to this [Intent](https://developer.android.com/reference/android/content/Intent.html).

Next, send the [Intent](https://developer.android.com/reference/android/content/Intent.html) by calling [LocalBroadcastManager.sendBroadcast()](https://developer.android.com/reference/android/support/v4/content/LocalBroadcastManager.html#sendBroadcast%28android.content.Intent%29).This sends the [Intent](https://developer.android.com/reference/android/content/Intent.html) to any component in your application that has registered to receive it.To get an instance of [LocalBroadcastManager](https://developer.android.com/reference/android/support/v4/content/LocalBroadcastManager.html), call [getInstance()](https://developer.android.com/reference/android/support/v4/content/LocalBroadcastManager.html#getInstance%28android.content.Context%29).

The next step is to handle the incoming broadcast [Intent](https://developer.android.com/reference/android/content/Intent.html) objects in the component that sent the original work request.

## 6.4.5 Receive Status Broadcasts from an IntentService

To receive broadcast [Intent](https://developer.android.com/reference/android/content/Intent.html) objects, use a subclass of [BroadcastReceiver](https://developer.android.com/reference/android/content/BroadcastReceiver.html). In the subclass, implement the [BroadcastReceiver.onReceive()](https://developer.android.com/reference/android/content/BroadcastReceiver.html#onReceive%28android.content.Context,%20android.content.Intent%29) callback method, which [LocalBroadcastManager](https://developer.android.com/reference/android/support/v4/content/LocalBroadcastManager.html) invokes when it receives an [Intent](https://developer.android.com/reference/android/content/Intent.html). [LocalBroadcastManager](https://developer.android.com/reference/android/support/v4/content/LocalBroadcastManager.html) passes the incoming [Intent](https://developer.android.com/reference/android/content/Intent.html) to [BroadcastReceiver.onReceive()](https://developer.android.com/reference/android/content/BroadcastReceiver.html#onReceive%28android.content.Context,%20android.content.Intent%29).

Once you've defined the [BroadcastReceiver](https://developer.android.com/reference/android/content/BroadcastReceiver.html), you can define filters for it that match specific actions, categories, and data.To do this, create an [IntentFilter](https://developer.android.com/reference/android/content/IntentFilter.html).To register the [BroadcastReceiver](https://developer.android.com/reference/android/content/BroadcastReceiver.html) and the [IntentFilter](https://developer.android.com/reference/android/content/IntentFilter.html) with the system, get an instance of [LocalBroadcastManager](https://developer.android.com/reference/android/support/v4/content/LocalBroadcastManager.html) and call its [registerReceiver()](https://developer.android.com/reference/android/support/v4/content/LocalBroadcastManager.html#registerReceiver%28android.content.BroadcastReceiver,%20android.content.IntentFilter%29) method.

 A single [BroadcastReceiver](https://developer.android.com/reference/android/content/BroadcastReceiver.html) can handle more than one type of broadcast [Intent](https://developer.android.com/reference/android/content/Intent.html) object, each with its own action. This feature allows you to run different code for each action, without having to define a separate [BroadcastReceiver](https://developer.android.com/reference/android/content/BroadcastReceiver.html) for each action.To define another [IntentFilter](https://developer.android.com/reference/android/content/IntentFilter.html) for the same [BroadcastReceiver](https://developer.android.com/reference/android/content/BroadcastReceiver.html), create the [IntentFilter](https://developer.android.com/reference/android/content/IntentFilter.html) and repeat the call to [registerReceiver()](http://developer.android.com/reference/android/support/v4/content/LocalBroadcastManager.html#registerReceiver%28android.content.BroadcastReceiver,%20android.content.IntentFilter%29).

Sending an broadcast [Intent](https://developer.android.com/reference/android/content/Intent.html) doesn't start or resume an [Activity](https://developer.android.com/reference/android/app/Activity.html).The [BroadcastReceiver](https://developer.android.com/reference/android/content/BroadcastReceiver.html) for an [Activity](https://developer.android.com/reference/android/app/Activity.html) receives and processes [Intent](https://developer.android.com/reference/android/content/Intent.html) objects even when your app is in the background, but doesn't force your app to the foreground. If you want to notify the user about an event that happened in the background while your app was not visible, use a [Notification](https://developer.android.com/reference/android/app/Notification.html). Never start an [Activity](https://developer.android.com/reference/android/app/Activity.html) in response to an incoming broadcast [Intent](https://developer.android.com/reference/android/content/Intent.html).

6.5 Finding the Location

We bring mobile devices almost everywhere.One of the unique features available to mobile applications is location awareness. Knowing the location and using the information wisely can bring a more contextual experience.

6.5.1 Using the Location Manager

Before the application can begin receiving location updates, it needs to perform some simple steps to set up access.

6.5.2 Declaring the proper permissions in android manifest

The first step of setting up location update access is to declare proper permissions in the manifest. If permissions are missing, the application will get a [SecurityException](http://developer.android.com/reference/java/lang/SecurityException.html) at runtime. Depending on the [LocationManager](http://developer.android.com/reference/android/location/LocationManager.html) methods used, either [ACCESS\_COARSE\_LOCATION](http://developer.android.com/reference/android/Manifest.permission.html#ACCESS_COARSE_LOCATION) or [ACCESS\_FINE\_LOCATION](http://developer.android.com/reference/android/Manifest.permission.html#ACCESS_FINE_LOCATION) permission is needed. For example, you need to declare the [ACCESS\_COARSE\_LOCATION](http://developer.android.com/reference/android/Manifest.permission.html#ACCESS_COARSE_LOCATION) permission if your application uses a network-based location provider only. The more accurate GPS requires the [ACCESS\_FINE\_LOCATION](http://developer.android.com/reference/android/Manifest.permission.html#ACCESS_FINE_LOCATION) permission. Note that declaring the [ACCESS\_FINE\_LOCATION](http://developer.android.com/reference/android/Manifest.permission.html#ACCESS_FINE_LOCATION) permission implies [ACCESS\_COARSE\_LOCATION](http://developer.android.com/reference/android/Manifest.permission.html#ACCESS_COARSE_LOCATION) already.

Also, if a network-based location provider is used in the application, you'll need to declare the internet permission as well.

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

## 6.5.3 Get a Reference to LocationManager

[LocationManager](http://developer.android.com/reference/android/location/LocationManager.html) is the main class through which the application can access location services on Android.Similar to other system services,a reference can be obtained from calling the [getSystemService()](http://developer.android.com/reference/android/content/Context.html#getSystemService%28java.lang.String%29) method. If your application intends to receive location updates in the foreground (within an [Activity](http://developer.android.com/reference/android/app/Activity.html)), you should usually perform this step in the [onCreate()](http://developer.android.com/reference/android/app/Activity.html#onCreate%28android.os.Bundle%29) method.

Manager =(LocationManager) this.getSystemService(Context.LOCATION\_SERVICE);

## 6.5.4 Pick a Location Provider

While not required, most modern Android-powered devices can receive location updates through multiple underlying technologies, which are abstracted to an application as [LocationProvider](http://developer.android.com/reference/android/location/LocationProvider.html) objects. Location providers may have different performance characteristics in terms of time-to-fix, accuracy, monetary cost, power consumption, and so on. Generally, a location provider with a greater accuracy, like the GPS, requires a longer fix time than a less accurate one, such as a network-based location provider.

Depending on your application's use case, you have to choose a specific location provider, or multiple providers, based on similar tradeoffs. For example, a points of interest check-in application would require higher location accuracy than say, a retail store locator where a city level location fix would suffice. The snippet below asks for a provider backed by the GPS.

Provider\_name =locationManager.getProvider(LocationManager.GPS\_PROVIDER);

Alternatively, you can provide some input criteria such as accuracy, power requirement, monetary cost, and so on, and let Android decide a closest match location provider. The snippet below asks for a location provider with fine accuracy and no monetary cost. Note that the criteria may not resolve to any providers, in which case a null will be returned. Your application should be prepared to gracefully handle the situation.

// Retrieve a list of location providers

List <String> l1=manager.getProviders(**true**);  
If no suitable provider is found, null is returned.

## 6.5.5 Verify the Location Provider is Enabled

Some location providers such as the GPS can be disabled in Settings.It is good practice to check whether the desired location provider is currently enabled by calling the [onProviderEnabled()](http://developer.android.com/reference/android/location/LocationManager.html#isProviderEnabled%28java.lang.String%29) method.

6.5.6 Obtaining the Current Location

After setting up an application to work with [LocationManager](http://developer.android.com/reference/android/location/LocationManager.html), you can begin to obtain location updates.

## 6.5.7 Set Up the Location Listener

The [LocationManager](http://developer.android.com/reference/android/location/LocationManager.html) class exposes a number of methods for applications to receive location updates. In its simplest form, you register an event listener, identify the location manager from which you'd like to receive location updates, and specify the minimum time and distance intervals at which to receive location updates.The [onLocationChanged()](http://developer.android.com/reference/android/location/LocationListener.html#onLocationChanged%28android.location.Location%29) callback will be invoked with the frequency that correlates with time and distance intervals.

6.6 Retrieving a List of Contacts

[Contacts Provider](http://developer.android.com/guide/topics/providers/contacts-provider.html) is the central repository of the user's contacts information, including data from contacts apps and social networking apps. In this app, we can access Contacts Provider information directly by calling [ContentResolver](http://developer.android.com/reference/android/content/ContentResolver.html) methods or by sending intents to a contacts app. This class focuses on retrieving lists of contacts, displaying the details for a particular contact, and modifying contacts using intents.

# 6.6.1 Accessing Contacts Data

We can retrieve a list of contacts whose data matches all or part of a search string, using the following techniques:

6.6.2 Match contact names

Retrieve a list of contacts by matching the search string to all or part of the contact name data. The Contacts Provider allows multiple instances of the same name, so this technique can return a list of matches.

6.6.3 Match a specific type of data, such as a phone number

Retrieve a list of contacts by matching the search string to a particular type of detail data such as an email address. For example, this technique allows you to list all of the contacts whose email address matches the search string.

6.6.4 Match any type of data

Retrieve a list of contacts by matching the search string to any type of detail data, including name, phone number, street address, email address, and so forth. For example, this technique allows you to accept any type of data for a search string and then list the contacts for which the data matches the string.

In this app we are using first technique.

6.6.5 Request permissions to read the provider.

To do any type of search of the Contacts Provider, your app must have [READ\_CONTACTS](http://developer.android.com/reference/android/Manifest.permission.html#READ_CONTACTS) permission. To request this, add this [<uses-permission>](http://developer.android.com/guide/topics/manifest/uses-permission-element.html) element to your manifest file as a child element of [<manifest>](http://developer.android.com/guide/topics/manifest/manifest-element.html):

<uses-permission android:name="android.permission.READ\_CONTACTS" />

6.6.6 Match a contact by a Name and list the results

This technique tries to match a search string to the name of a contact or contacts in the Contact Provider's [ContactsContract.Contacts](http://developer.android.com/reference/android/provider/ContactsContract.Contacts.html) table. You usually want to display the results in a [ListView](http://developer.android.com/reference/android/widget/ListView.html), to allow the user to choose among the matched contacts.

6.6.7 Define constants for the Cursor column indexes

To get data from an individual column in a [Cursor](http://developer.android.com/reference/android/database/Cursor.html), you need the column's index within the [Cursor](http://developer.android.com/reference/android/database/Cursor.html).

6.6.8 Specify the selection criteria

To specify the data you want, create a combination of text expressions and variables that tell the provider the data columns to search and the values to find.

6.7 Retrieve the call history

To get call history first add read contact permission in Manifest file :  
<uses-permission android: name="android.permission.READ\_CONTACTS" />

Working with the Android Call Logs is really pretty simple. Basically, a content provider is a special kind of database that Android uses so that programs can expose their data to each other. To use a content provider, you just need to do is call the "getContentResolver ()" method. Once you have a content resolver object, you can query it using SQL or any of the built in query functions. One such Content Provider is android.provider.CallLog.Calls which provides access to the call logs data. Call Logs contain information about outgoing, incoming and missed calls.

String phNumber = managedCursor.getString ( number );

String call Type = managedCursor.getString ( type );

String callDate =managedCursor.getString ( date );

Date callDayTime = **new** Date(Long.valueOf(callDate));

String callDuration = managedCursor.getString( duration );

The following steps show how to access the call log history:

1. Get the URI of the call log content provider.

2. Query the content provider for specific information.

3. Read data through the use of Cursor.

##### 6.7.1 Get the URI of the Call Log Content Provider

android.provider.CallLog.Calls is a static class that exposes a CONTENT\_URI field for the call log content provider (database). The CONTENT\_URI field indicates the table from where the call logs data would be access. CONTENT\_URI will be passed to the query method as explained in step 2.

6.7.2 Query the Content Provider for Specific Information.

The android.content.ContentResolver query() method is used to query the content provider.Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder). Querying the given URI returns a Cursor over the result set. Parameters of the query(…) method are:

* **uri** - will be android.provider.CallLog.Calls (Where to get data from ? )
* **projection** - will be the columns to be return. Passing null will return all columns of the content provider (What columns to return?)
* **sortOrder** - the sorting order of the rows (How to sort the rows?.)

The following table presents the android.provider.CallLog.Calls columns constants that can be used for projections:

|  |  |
| --- | --- |
| CACHED\_NAME (String) | The cached name associated with the phone number, if it exists. |
| CACHED\_NUMBER\_LABEL (String) | The cached number label for a custom number type associated with the phone number, if it exists. |
| CACHED\_NUMBER\_TYPE (String) | The cached number type (Home, Work, etc) associated with the phone number, if it exists. |
| DATE (String) | The date the call occurred in milliseconds since an epoch |
| DURATION (String) | The duration of the call in seconds |
| IS\_READ (String) | Whether an item is read or consumed by the user. |
| NEW (String) | Whether or not the call has been acknowledged |
| NUMBER (String) | The phone number as the user entered it. |
| TYPE (String) | The type of the call (incoming, outgoing or missed). |

The following code snippet shows the use of query method to access call logs from content provider.

ContentResolver cr = getContentResolver();

/\*method present in android.content.Contextclass\*/

Cursor managedCursor = cr.query(android.provider.CallLog.Calls.CONTENT\_URI, /\*uri\*/

null, /\*projection\*/

null,/\*selection\*/

null,/\*selection arguments\*/

android.provider.CallLog.Calls.DEFAULT\_SORT\_ORDER /\*sortby\*/);

##### 6.7.3 Read data through the use of android.database.Cursor

Cursor provides read access to the result set returned by the query() method. To read the data loop through the result set..

## 6.8 Telephony Manager

## Telephony Manager provides access to information about the telephony services on the device. Applications can use the methods in this class to determine telephony services and states, as well as to access some types of subscriber information.Applications can also register a listener to receive notification of telephony state changes.

## You do not instantiate this class directly; instead, you retrieve a reference to an instance through Context.getSystemService(Context.TELEPHONY\_SERVICE).

## 6.8.1 Permission Required*:*

## To work with Telephony Manager and to read the phone details we need <uses-permission android:name="android.permission.READ\_PHONE\_STATE"/>           permission.So add this permission in the manifest file.

## 6.8.2 Accessing the telephony manager

## TelephonyManager telephonyManager=(TelephonyManager)getSystemService(Context.TELEPHONY\_SERVICE); Using the Object of Telephony Manager class we can get the details like SIM Serial number, Country code Network provider code and other details.

## 6.9 AudioManager

Instead of using a **ContentResolver** to obtain the service that controls the audio, Android has a class that makes everything easier, named **AudioManager**. So, the example here will basically obtain a reference to the **AudioManager** and will use it to get the current ringer volume and use it to set the progress of a seek bar. When changed, the seek bar progress will set the value of the sound volume. Here is the code:

Set ringer to silent mode

public void setRinger2Silent()

{

AudioManager audioManager= (AudioManager) getBaseContext().getSystemService(Context.AUDIO\_SERVICE);

am.setRingerMode(AudioManager.RINGER\_MODE\_SILENT);

}

Set silent to ringer mode

public void setRinger2Silent()

{

AudioManager audioManager= (AudioManager) getBaseContext().getSystemService(Context.AUDIO\_SERVICE);

am.setRingerMode(AudioManager.RINGER\_MODE\_NORMAL);

}

6.10 Accessing Inbox Messages

## Permission Required*:*

<uses-permission android:name="android.permission.READ\_SMS"></uses-permission>

In Android we can Access all the SMS stored in Inbox.we can also fetch the Sender Number, Data, Time ,SMS Body etc.

We can fetch the SMS from Inbox by making query to SMS content resolver by using:

Cursor cursor = getContentResolver().query(Uri.parse("content://sms/inbox"), null, null, null, null);  
sender Number=cursor.getString(cursor.getColumnIndex("address"));  
String smsBody=cursor.getString(cursor.getColumnIndex("body"));

**Chapter 7**

**TESTING**

The primary goal of testing is to determine if software is as desired, that is it conforms to the requirements. Testing focuses on semantic errors also known as faults that causes the program under test to behave incorrectly. Testing is vital to the success of the system. Software testing makes a logical assumption whether all parts of the system are correct and whether the goal is successfully achieved. The candidate system is subjected to variety of tests online. A series of tests are performed before the system is ready for user acceptance.

7.1 Unit testing

During this stage, the software design is realized as a set of program or program unit. Unit testing involves verifying that each unit meets the specification.

7.2 Test Case

Test case includes input data, execution conditions and pass/fail criteria, that is it consists of the program input and expected output.

7.2.1 PIN entering activity

|  |  |  |  |
| --- | --- | --- | --- |
| S.NO | Input Variable | Expected Result | Actual Result |
| 1. | PIN not entered | Error Message(PIN should be entered) | Displays an Error Message |

7.2.2 Update PIN Activity

|  |  |  |  |
| --- | --- | --- | --- |
| S.NO | Input Variable | Expected Result | Actual Result |
| 1. | New PIN, Confirm PIN only | Error Message(Old PIN should be entered) | Displays an error message |
| 2. | Old PIN, New PIN only | Error Message(Confirm PIN should be entered) | Displays an error message |
| 3. | Old PIN, Confirm PIN only | Error Message(New PIN should be entered) | Displays an error message |
| 4. | Incorrect Old PIN | Error Messag(Incorrect Old PIN) | Displays an error message |
| 5. | Incorrect New PIN and Confirm PIN | Error Message (New PIN and Confirm PIN do not match) | Displays an error message |

Manual Testing

The application was tested manually that is we installed the application in the smartphone and tested for its features.

After testing it was found that the application works as desired for all the options and all the objectives of our project were achieved.

Fig 8.26 Message showing status of connection

**Conclusion and Future Scope**

The proposed model facilitates accessing of the device from a remote location.

The system has been designed in such a way that the mobile terminal used for accessing a remote android device need not be an android device.

By using this application, we can retrieve the data stored in a smartphone. We can retrieve Contacts (Fetching contact number from your address book), Call Logs (Checking missed calls/received calls/dialed numbers), IMEI Number, Phone Profile (Changing profile mode to ringer or silent mode), SIM Number (Retrieving SIM Number), SMS Logs (Checking SMS’s received on your phone), SIMOP,Location of the device.

Future directions of this work includes

* The remote connection through SMS can be replaced by GPRS. General packet radio service (GPRS) is a [packet oriented](http://en.wikipedia.org/wiki/Packet_oriented) [mobile data service](http://en.wikipedia.org/wiki/Mobile_Data_Service) on the [2G](http://en.wikipedia.org/wiki/2G) and [3G](http://en.wikipedia.org/wiki/3G) [cellular communication](http://en.wikipedia.org/wiki/Cellular_communication) system's [global system for mobile communications](http://en.wikipedia.org/wiki/Global_System_for_Mobile_Communications) (GSM).If SMS over GPRS is used, an SMS transmission speed of about 30 SMS messages per minute may be achieved. This is much faster than using the ordinary SMS over GSM, whose SMS transmission speed is about 6 to 10 SMS messages per minute
* The screen capturing of remote device can be incorporated so that the exact display can be accessed

**USER MANUAL**

**Installation procedure**

Step 1: The first step in installing the application is to copy the file with .apk extension that is stored in bin folder of the software.

Step 2: After copying the apk file, click on it, then a screen that contains the details of the facilities provided by the application and also the options to install and quit appears.

Step 3: Click on install if you want to install the app or quit if you want to exit. If you click on install the next screen appears, here we have options like package installer and go back, in case if you don’t want to install the app click on go back, otherwise click on package installer.

Step 4: After clicking on the installer, if the installation is done, we get a notification stating application installed successfully, otherwise a failure notification is sent stating application not installed, in this case it is recommended to repeat the above procedure.

**Usage**

Step 1: Once the installation is done, an icon with the label mobile access appears on the screen.

Step 2: Click on that icon.

Step 3: A screen asking to input the pin appears, enter the pin of your choice, this pin will be used further to establish the connection with the device remotely.

Step 4: After the pin is created, the screen containing an image and a button called change pin will appear, this will be the main screen of the app. In case if you want to change the pin click on the button.

Step 5: Once the button is clicked you get the screen asking you to enter the old pin, new pin and confirm pin, if any of these fields is left empty, or if the old pin entered is not correct or if the new pin and confirm pin do not match, you get a failure message respectively, otherwise you get a message saying pin updated successfully.

Step 6: In order to retrieve the data from the phone remotely, first send a remote connection command with the pin that you have stored during installation. Ex if you have a pin called qwerty, you are supposed to send a message like “PIN qwerty” to your phone

Step 7: wait for the app to return a list of options, once you get the list, select the option you want and send it as a message to your phone. Ex if you want to retrieve the IMEI number, send a message like IMEI. The application will run in the background and will return the result for the selected option.

**References**

[1] Android Apps For Absolute Beginners byWallace Jackson, 2011

[2] Survey about mobile theft in UK: http://news.bbc.co.uk/2/hi/uk\_news/1748258.stm

[3] Survey about mobile theft in USA:

http://www.symantec.com/about/news/release/article.jsp?prId=20110208\_01

[4] Survey about mobile theft in India:http://asiarelease.asia/norton-survey-reveals-1-in-2-indians-is-avictim-ofmobile-phone-loss-or-theft/

[5] Semantec Anti theft: https://market.android.com/details?id=com.symantec.anti.theft

[6] Mcafee Wave secure: https://market.android.com/details?id=com.wsandroid

[7] Android SDK: http://developer.android.com/sdk/android-2.3.html

[8] AndroidEmulator:http://developer.android.com/guide/developing/tools/emulator.html