



Jaypee University of Information Technology
Solan (H.P.)
LEARNING RESOURCE CENTER

Acc. Num. **SP12014** Call Num:

General Guidelines:

- ◆ Library books should be used with great care.
- ◆ Tearing, folding, cutting of library books or making any marks on them is not permitted and shall lead to disciplinary action.
- ◆ Any defect noticed at the time of borrowing books must be brought to the library staff immediately. Otherwise the borrower may be required to replace the book by a new copy.
- ◆ The loss of LRC book(s) must be immediately brought to the notice of the Librarian in writing.

Learning Resource Centre-JUIT



SP12014

REAL TIME USER MONITORING USING ANDROID SMARTPHONE BASED APP

Project report submitted in partial fulfillment of the requirement for the degree of
Bachelor of Technology

in

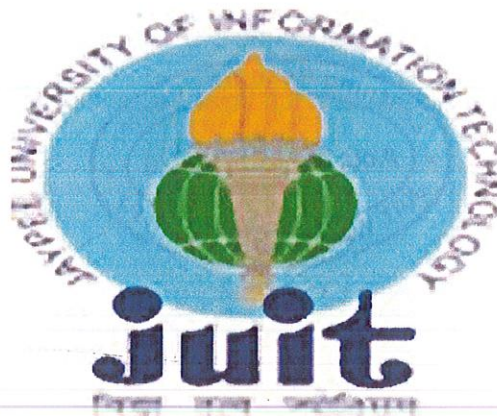
Computer Science and Engineering
By

Tushar Dhembala(121323)
Abhinav Kanaujia(121232)

Under the supervision of

Dr. Shailendra Shukla

to



Department of Computer Science & Engineering and Information Technology
**Jaypee University of Information Technology Waknaghat, Solan-173234,
Himachal Pradesh**



Candidate's Declaration

I hereby declare that the work presented in this report entitled "**REAL TIME USER MONITORING USING ANDROID SMARTPHONE BASED APP**" in fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from August 2015 to May 2016 under the supervision of **Dr. Shailendra Shukla** (Dept. of Computer Science and Engineering).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

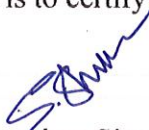


(Student Signature)
Tushar Dhembala, 121323



(Student Signature)
Abhinav Kanaujia, 121232

This is to certify that the above statement made by the candidate is true to the best of my knowledge.



(Supervisor Signature)
Dr. Shailendra Shukla
Assistant Professor (Senior Grade)
Computer Science and Engineering

Dated: 31/5/16

ACKNOWLEDGEMENT

This project work would not have been possible without the guidance and the help of several individuals who in one way or another contributed and extended their valuable assistance in the preparation and completion of this study.

First and foremost I offer my sincerest gratitude to my supervisor **Dr. Shailendra Shukla** who has supported me throughout my project work with his patience and knowledge whilst allowing me the room to work in my own way. I attribute the level of my Masters degree to her encouragement and effort and without him this thesis, too, would not have been completed or written. One simply could not wish for a better or friendlier supervisor.

The work was performed at Department of Computer Science and Engineering at the University, and I would like to thank all the people there for their hospitality and support.

Last, but not the least, my family and the one above all of us, the omnipresent God.

Date:

Tushar dhembla
(121323)

Abhinav kanaujia
(121232)

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION	1-19
1.1 INTRODUCTION	1
1.2 OBJECTIVES	2
1.3 MODULAR DESCRIPTION OF THE JOB	3
1.4 USERS OF THE SYSTEM	4
1.5 INDUSTRY APPLICATION	4

CHAPTER 2

LITERATURE SURVEY	6-14
2.1 EXISTING LITERATURE	6
2.2 SIMILAR ANDROID APPS ON SECURITY AND MONITORING	10

CHAPTER 3

SYSTEM DEVELOPMENT	15-23
3.1 HARDWARE REQUIREMENTS	15
3.2 SOFTWARE REQUIREMENTS	15
3.3 LOCATION TRACKING	16
3.4 FUNCTIONAL REQUIREMENTS	18
3.5 SYSTEM REQUIREMENTS	20

CHAPTER 4

RESULTS AND DISCUSSION	24-37
4.1 INTERFACE	24
4.2 COUNTING DOWN	27

CHAPTER 6
CONCLUSION

38

REFERENCES / BIBLIOGRAPHY

39

LIST OF FIGURES

Figure 1.1 - Android's architecture diagram	12
Figure 1.2 - The first-generation Nexus 7 tablet, running Android 4.1 Jelly Bean	17
Figure 1.3 – Screenshot App	24
Figure 1.4 – ScreenGrabber App	25
Figure 1.5 - Screenshot Snap Free	26
Figure 1.6 – CaptureScreen	27
Figure 1.7 – Take Screenshot App	28
Figure 4.1 – App Activity	30
Figure 4.2 – Android sdcard explorer view	32
Figure 4.3 – Timer in the App	33
Figure 4.3 – Screenshot Taking Execution	34

ABSTRACT

In this work the android app development for parents and security agencies is developed. In this implementation, the development of a background app will be covered to take the screenshots automatically from Mobile Handset and sending to a particular E-mail ID. This app is useful for the monitoring of users (For Parents, Security Agencies) about their activities on the mobile phone. The same screenshots can be uploaded to a cloud platform so that the logs can be checked. Here in this work the complete code is explained for whole activity and background service. In this app we are capturing a screenshot of the activity by `getRootView` method then we are compressing it to a PNG or JPG format which is further saved in the external files directory and emailed to the appropriate receiver. In this process it is suggested that do not save the bitmap/screenshot in the internal files directory as if we do that our native phone's email app won't be able to access it, as android does not allow applications to access files local to other applications, doing so wont throw an error but as a result- image will not be attached to the email.

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

These, days, there is no stopping a child from growing up without the influence of tech gadgets, online content, and the Internet in general. Being a parent is already a tough enough task, but shying away from the topic or completely shutting out technology from your children's lives isn't going to help. What you can do however, is to have more control over what they are exposed to while on their gadgets. Various techniques are being carried out by employers to monitor the work, one of the common way is CCTV cameras. But when employees are not on the same location then CCTV cameras installation is costly enough. To take this point in mind an android application has been designed.



In this work the android app development for parents and security agencies is developed. In this implementation, the development of a background app will be covered to take the screenshots automatically from Mobile Handset and sending to a particular E-mail ID. This app is useful for the monitoring of

users (For Parents, Security Agencies) about their activities on the mobile phone. The same screenshots can be uploaded to a cloud platform so that the logs can be checked. Here in this

work the complete code is explained for whole activity and background service. In this app we are capturing a screenshot of the activity by `getRootView` method then we are compressing it to a PNG or JPG format which is further saved in the external files directory and emailed to the appropriate receiver. In this process it is suggested that do not save the bitmap/screenshot in the internal files directory as if we do that our native phone's email app won't be able to access it, as android does not allow applications to access files local to other applications, doing so wont throw an error but as a result- image will not be attached to the email.

1.2 OBJECTIVES

In this application, application can take screen shots of the work which is being processed by the kids from the various locations.

In this system, administrator or we can say parents have all the rights to manage various employees(kids) information, he has also rights to start the server or monitor the work by various employees.

The screen shots process will automatically start once the parents start the time on the app and they will be saved in sd card or will be sent on the parents id with location.

When employee want to sign out from the grabbing application then all the work will be uploaded to the server machine for monitoring.

The objective of the Server(parent) system would be to:

1. Keep the information of all employees(kids).
2. Get the employee activity information.
3. Generate various reports like user activity.

The objective of the Client(kid) system would be to:

1. Generate and Manipulate information of all the employees.
2. Generate employee activity information.
3. Taking snapshots of the activities of the employees.

1.3 MODULAR DESCRIPTION OF THE JOB

- Input Requirements of the System
 1. Login.
 2. User Details
 3. Manage profile information
 4. Start the Server

- **Output Requirements of the System**

1. Listing of all Users.
2. Listing of all User Activity.

1.4 USERS OF THE SYSTEM

The users of this system will be the users of the organization. The systems are menu driven to facilitate the users. The system is developed with the participation of users, which will help them to understand the system easily.

1.4.1 Administrator: User of type administrator will manage the administration part of the application. Also, he is allowed to manage all the users exist in the system.

1.4.2 Employee: User of type employee is the user who will be under Surveillance of Administrator.

1.5 INDUSTRY APPLICATION

Various techniques are being carried out by employers to monitor the work, one of the common way is CCTV cameras. But when employees are not on the same location then CCTV cameras installation is costly enough. To take this point in mind AN ANDROID APP has been designed.

1.5.1 REAL TIME CAPTURING

In this application, application can take screen shots of the work which is being processed by various employees from the various locations. This application greatly provides with the facility of remotely capturing the activities of the employees on various client machines irrespective of the location of the client machines.

1.5.2 NETWORK SECURITY

This application deals with providing logical security, access controls and auditing systems as well.

In this system, administrator has all the rights to manage various employees' information, he has also rights to start the server or monitor the work by various Employees by establishing a client server network connection.

Whereas, employee has the rights to manage their own profile. The screen shots process will automatically start after the login by the employee and they will be saved on the client machine.

CHAPTER 2

LITERATURE SURVEY

2.1 EXISTING LITERATURE

Enck, W., Gilbert, P., Han, S., Tendulkar, V., Chun, B. G., Cox, L. P., ... & Sheth, A. N. (2014). TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Transactions on Computer Systems (TOCS)*, 32(2), 5.

Today's smartphone operating systems frequently fail to provide users with visibility into how third-party applications collect and share their private data. We address these shortcomings with TaintDroid, an efficient, system-wide dynamic taint tracking and analysis system capable of simultaneously tracking multiple sources of sensitive data. TaintDroid enables realtime analysis by leveraging Android's virtualized execution environment. TaintDroid incurs only 32% performance overhead on a CPU-bound microbenchmark and imposes negligible overhead on interactive third-party applications. Using TaintDroid to monitor the behavior of 30 popular third-party Android applications, in our 2010 study we found 20 applications potentially misused users' private information; so did a similar fraction of the tested applications in our 2012 study. Monitoring the flow of privacy-sensitive data with TaintDroid provides valuable input for smartphone users and security service firms seeking to identify misbehaving applications.

Backes, M., Gerling, S., Hammer, C., Maffei, M., & von Styp-Rekowsky, P. (2013). AppGuard—enforcing user requirements on android apps. In *Tools and Algorithms for the Construction and Analysis of Systems* (pp. 543-548). Springer Berlin Heidelberg.

The success of Android phones makes them a prominent target for malicious software, in particular since the Android permission system turned out to be inadequate to protect the user against security and privacy threats. This work presents *AppGuard*, a powerful and flexible system for the enforcement of user-customizable security policies on untrusted Android applications. AppGuard does not require any changes to a smartphone's firmware or root access. Our system offers complete mediation of security-relevant methods based on callee-site inline reference monitoring. We demonstrate the general

applicability of AppGuard by several case studies, e.g., removing permissions from overly curious apps as well as defending against several recent real-world attacks on Android phones. Our technique exhibits very little space and runtime overhead. AppGuard is publicly available, has been invited to the Samsung Apps market, and has had more than 500,000 downloads so far.

Heuser, S., Nadkarni, A., Enck, W., & Sadeghi, A. R. (2014). Asm: A programmable interface for extending android security. In 23rd USENIX Security Symposium (USENIX Security 14) (pp. 1005-1019).

Android, iOS, and Windows 8 are changing the application architecture of consumer operating systems. These new architectures required OS designers to rethink security and access control. While the new security architectures improve on traditional desktop and server OS designs, they lack sufficient protection semantics for different classes of OS customers (e.g., consumer, enterprise, and government). The Android OS in particular has seen over a dozen research proposals for security enhancements. This paper seeks to promote OS security extensibility in the Android OS. We propose the Android Security Modules (ASM) framework, which provides a programmable interface for defining new reference monitors for Android. We drive the ASM design by studying the authorization hook requirements of recent security enhancement proposals and identify that new OSes such as Android require new types of authorization hooks (e.g., replacing data). We describe the design and implementation of ASM and demonstrate its utility by developing reference monitors called ASM apps. Finally, ASM is not only beneficial for security researchers. If adopted by Google, we envision ASM enabling in-the-field security enhancement of Android devices without requiring root access, a significant limitation of existing bring-your-own-device solutions.

Barrera, D., Clark, J., McCarney, D., & van Oorschot, P. C. (2012, October). Understanding and improving app installation security mechanisms through empirical analysis of android. In Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices (pp. 81-92). ACM.

This work provide a detailed analysis of two largely unexplored aspects of the security decisions made by the Android operating system during the app installation process: update integrity and UID assignment. To inform our analysis, we collect a dataset of Android application metadata and extract features from these binaries to gain a better understanding of how developers interact with the security mechanisms invoked during installation. Using the dataset, we find empirical evidence that Android's current signing architecture does not encourage best security practices. We also find that limitations of Android's UID sharing method force developers to write custom code rather than rely on OS-level mechanisms for secure data transfer between apps. As a result of our analysis, we recommend incrementally deployable improvements, including a novel UID sharing mechanism with applicability to signature-level permissions. We additionally discuss mitigation options for a security bug in Google's Play store, which allows apps to transparently obtain more privileges than those requested in the manifest.

Ongtang, M., McLaughlin, S., Enck, W., & McDaniel, P. (2012). Semantically rich application-centric security in Android. *Security and Communication Networks*, 5(6), 658-673.

Smartphones are now ubiquitous. However, the security requirements of these relatively new systems and the applications they support are still being understood. As a result, the security infrastructure available in current smartphone operating systems is largely underdeveloped. In this paper, we consider the security requirements of smartphone applications and augment the existing Android operating system with a framework to meet them. We present Secure Application INteraction (Saint), a modified infrastructure that governs install-time permission assignment and their run-time use as dictated by application provider policy. An in-depth description of the semantics of application policy is presented. The architecture and technical detail of Saint are given, and areas for extension, optimization, and improvement are explored. We demonstrate through a concrete example and study of real-world applications that Saint provides necessary utility for applications to assert and control the security decisions on the platform. Copyright © 2011 John Wiley & Sons, Ltd.

Russello, G., Jimenez, A. B., Naderi, H., & van der Mark, W. (2013, December). FireDroid: hardening security in almost-stock Android. In Proceedings of the 29th Annual Computer Security Applications Conference (pp. 319-328). ACM.

Malware poses a serious threat to Android smartphones. Current security mechanisms offer poor protection and are often too inflexible to quickly mitigate new exploits. In this paper we present FireDroid, a policy-based framework for enforcing security policies by interleaving process system calls. The main advantage of FireDroid is that it is completely transparent to the applications as well as to the Android OS. FireDroid enforces security policies without modifying either the Android OS or its applications. FireDroid is able to perform security checks on third-party and pre-installed applications, as well as malicious native code. We have implemented a novel mechanism that is able to attach, identify, monitor and enforce policies for any process spawned by the Android's mother process Zygote. The authors have tested the effectiveness of FireDroid against real malware. Moreover, we show how FireDroid can be used as a swift solution for blocking OS and application vulnerabilities before patches are available. Finally, we provide an experimental evaluation of our approach showing that it has only a limited overhead. Given these facts, FireDroid represents a practical solution for strengthening security on Android smartphones.

Chen, Q. A., Qian, Z., & Mao, Z. M. (2014). Peeking into your app without actually seeing it: UI state inference and novel android attacks. In 23rd USENIX Security Symposium (USENIX Security 14) (pp. 1037-1052).

The security of smartphone GUI frameworks remains an important yet under-scrutinized topic. In this paper, the authors report that on the Android system (and likely other OSes), a weaker form of GUI confidentiality can be breached in the form of UI state (not the pixels) by a background app without requiring any permissions. Our finding leads to a class of attacks which we name *UI state inference attack*. The underlying problem is that popular GUI frameworks by design can potentially reveal every UI state change through a newly-discovered public side channel — shared memory. In our evaluation, we show that for 6 out of 7 popular Android apps, the UI state inference accuracies are 80–90% for

the first candidate UI states, and over 93% for the top 3 candidates. Even though the UI state does not reveal the exact pixels, we show that it can serve as a powerful building block to enable more serious attacks. To demonstrate this, we design and fully implement several new attacks based on the UI state inference attack, including hijacking the UI state to steal sensitive user input (*e.g.*, login credentials) and obtain sensitive camera images shot by the user (*e.g.*, personal check photos for banking apps). We also discuss non-trivial challenges in eliminating the identified side channel, and suggest more secure alternative system designs.

2.2 SIMILAR ANDROID APPS ON SECURITY AND MONITORING

Screenshot

This app does allow for a variety of ways to take a screenshot. However, more importantly, the app allows for users to dictate how the screenshots are used afterwards. You can specify where they are saved, edit the images, add text or other designs and generally jazz up the images.

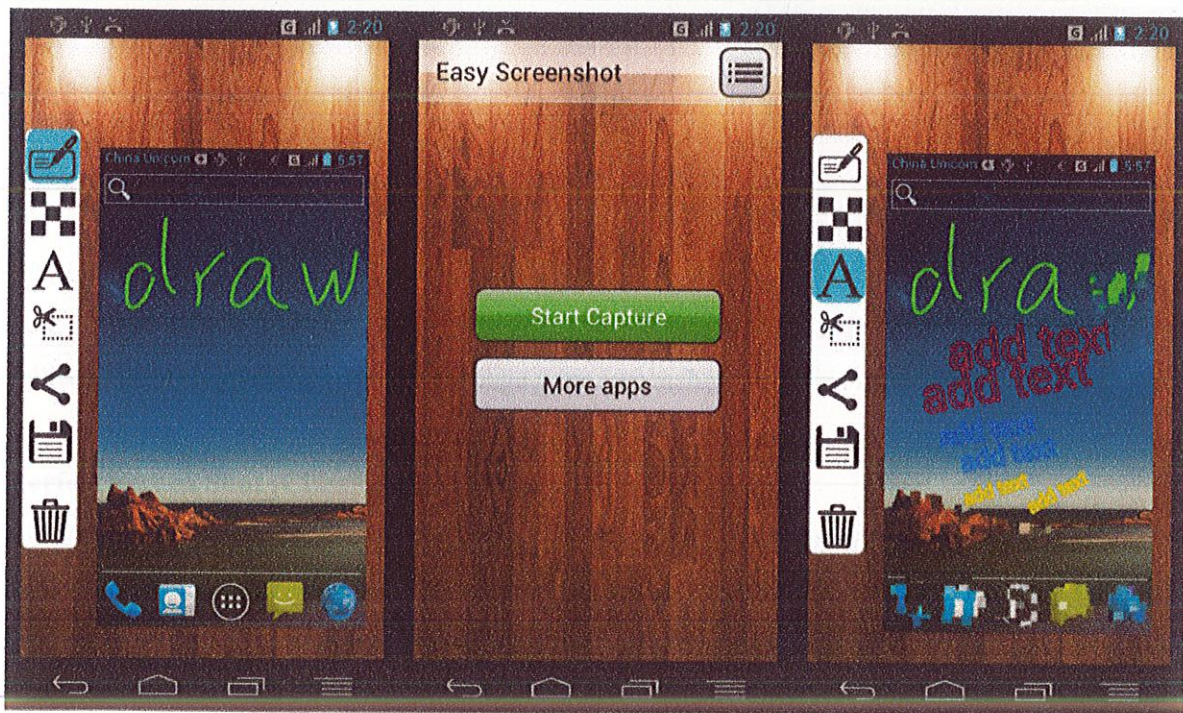


Figure 1.3 – Screenshot App

Screen Grabber



Figure 1.4 – ScreenGrabber App

Screen Grabber is an app which focuses more on the sharing ability of screenshots. Although, this can be largely done quite easily now on newer smartphones, if you would prefer an app which can assist and help to better manage your screenshots then this might be the options to try.

Perfect Screen Shot (Classic)

Once again, this app is more focused on what you can do with your screenshots more so than the actual act of taking a screenshot. Not to mention, this app contains the ability to apply your screenshots to the front of various phone frames. Perfect if you are planning on releasing your own app soon.

Screenshot Snap Free

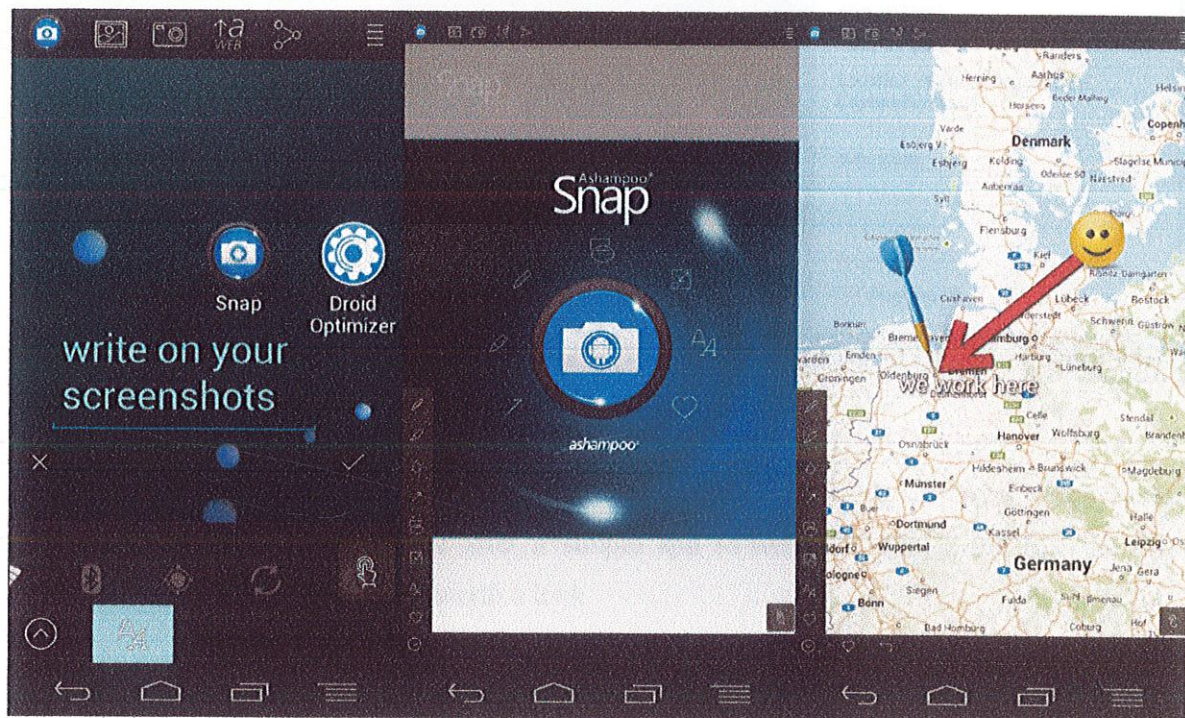


Figure 1.5 - Screenshot Snap Free

Screenshot Snap is another app which places its focus on the ability to edit screenshots after they have been taken. With this one, you can annotate shots, add text or further customize the images. Not to mention, an increased compatibility to share your images all from within the app.

Capture Screen

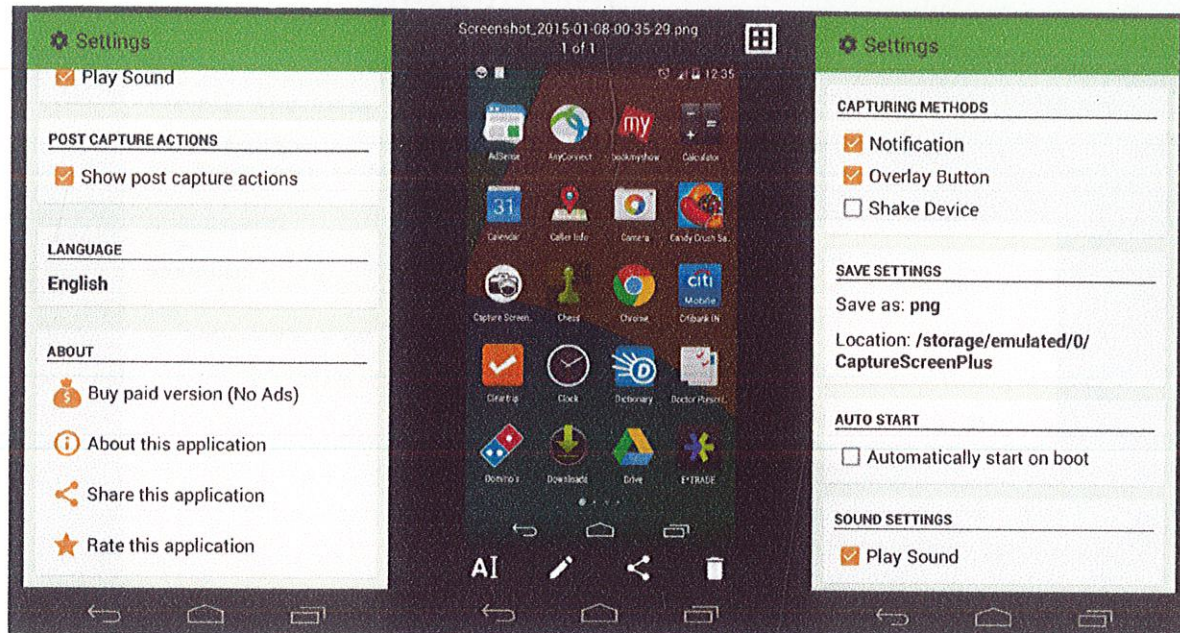


Figure 1.6 - CaptureScreen

Compared to some of the other options, Capture Screen is a little more basic and offers limited functionality. However, if you are after a simple app to help you group and manage your screenshots then this might be one worth a look

Take Screenshot

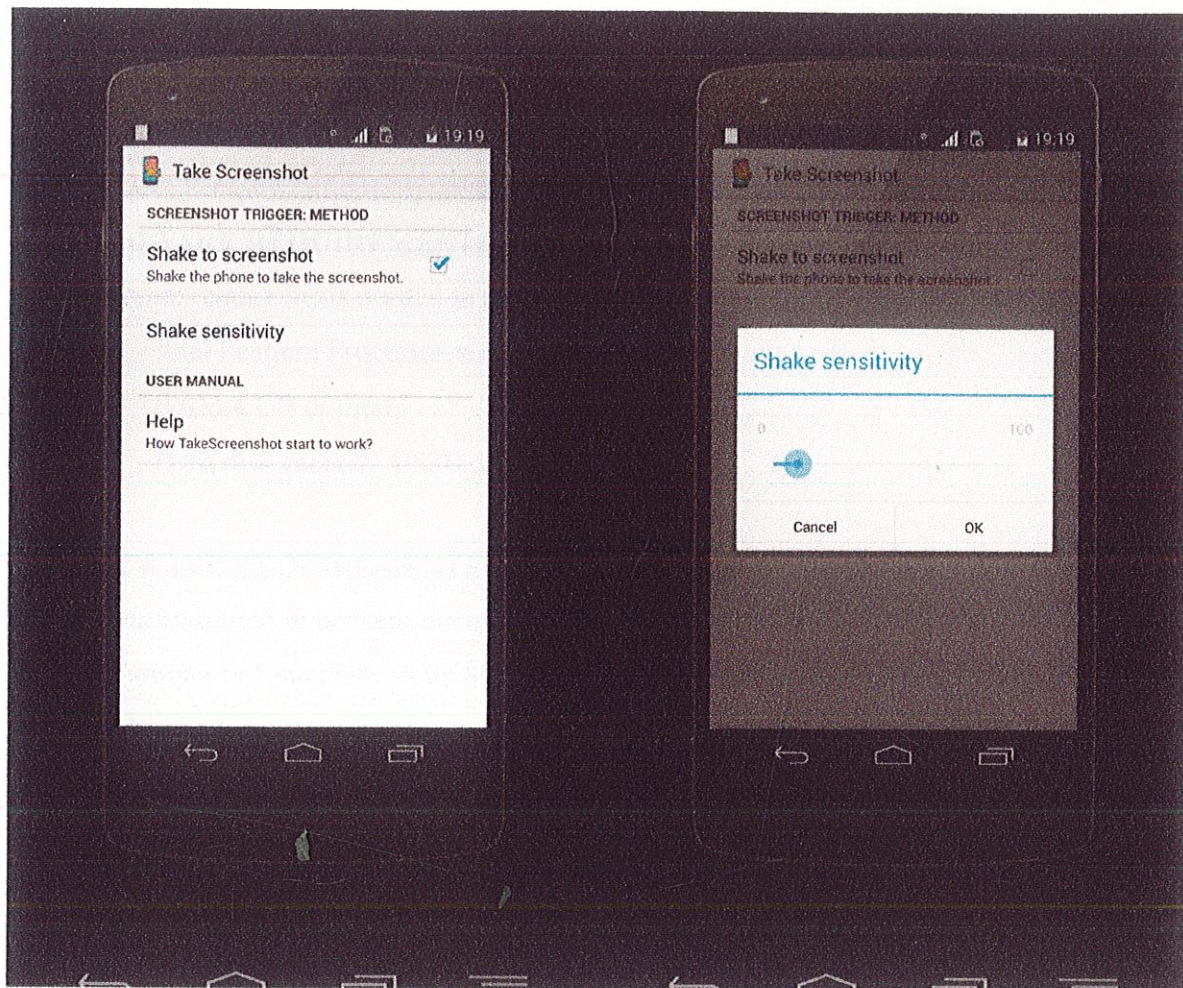


Figure 1.7 – Take Screenshot App

Take Screenshot offers a bit more of an interactive method of taking your shots. As such, this app will allow users the ability to take a screenshot by simply shaking their device. Not to mention, the degree of shaking needed can be adjusted and tweaked. Good for those who do not like the more traditional hold power and volume down options.

CHAPTER 3

SYSTEM DEVELOPMENT

3.1 HARDWARE REQUIREMENTS

The minimum requirements needed to perform operations are

- Intel Pentium Processor at 2 GHz or Higher
- RAM 4 GB or more
- Hard disk capacity 10GB or more

3.2 SOFTWARE REQUIREMENTS

The software required to perform the implementation are

- Windows or Linux Operating System (Ubuntu, Fedora)
- Advance Java
- Android Studio 2
- Notepad++
- Android Emulator

3.3 LOCATION TRACKING OF ANDROID DEVICE

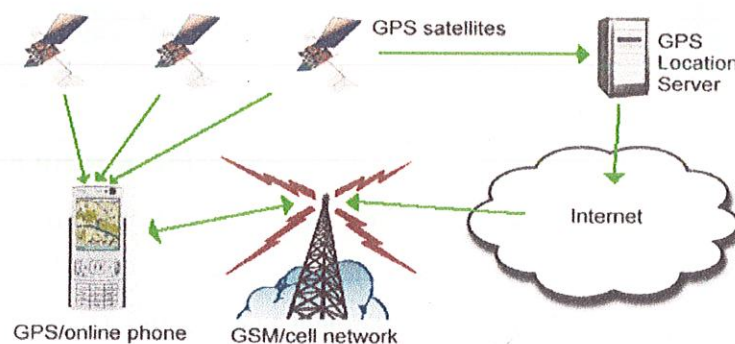
-

- **Figure 2.2: Structure of Android Components**

- **Activity**
 - An Activity is, fundamentally, an object that has a lifecycle. An Activity is a chunk of code that does some work; if necessary, that work can include displaying a UI to the user. It doesn't have to, though-some Activities never display UIs. Typically, we will designate one of our application's Activities as the entry point to our application.
 - **Broadcast Receiver**
 - Broadcast Receiver is yet another type of component that can receive and respond to any broadcast announcements.
 - **Service**
 - A Service is a body of code that runs in the background. It can run in its own process, or in the context of another application's process, depending on its needs. Other components "bind" to a Service and invoke methods on it via remote procedure calls. An example of a Service is a media player; even when the user quits the media-selection UI, she probably still intends for her music to keep playing. A Service keeps the music going even when the UI has completed.
 - **Content Provider**
 - Content Provider is a data storehouse that provides access to data on the device; the classic example is the Content Provider that's used to access the user's list of contacts. Our application can access data that other applications have exposed via a Content Provider, and we can also define our own Content Providers to expose data of our own.
-
- **2.3.2.3 Location based Services in Android**
 - Android's Network Location Provider determines user location using cell tower and Wi-Fi signals, providing location information in a way that works indoor and outdoor, responds faster, and uses less battery power. The purpose of location-based services is to

find the Physical location of the device. Access to the location-based services is handled by the LocationManager system Service. To access the Location Manager, request an instance of the LOCATION_SERVICE using the get System Service() method. Current Location can be fetched using two ways:

- 1. GPS (Global Positioning System)
- 2. Network Service Location
-
- **GPS (Global Positioning System)**
- The Global Positioning System (GPS) uses a constellation of 24 satellites orbiting the earth. GPS finds the user position by calculating differences in the times the signals, from different satellites, take to reach the receiver. GPS signals are decoded, so the smart phone must have in-built GPS receiver. To get access to GPS hardware of android we request using following statement **LocationManager.GPS_PROVIDER;**

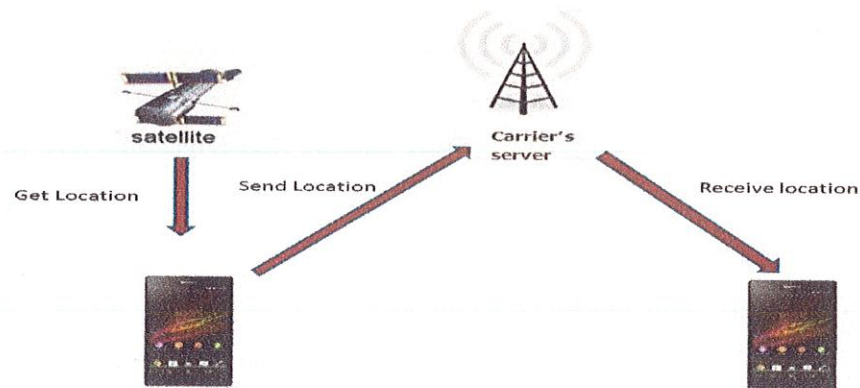


• **Figure 2.3: Architecture of A-GPS System**

- **Network Service Location**
- The current cell ID is used to locate the Base Transceiver Station (BTS) that the mobile phone is interacting with and the location of that BTS. It is the most basic and cheapest

method for this purpose as it uses the location of the radio base station that the cell phone is connected to. A GSM cell may be anywhere from 2 to 20 kilometers in diameter. Other approaches used along with cell ID can achieve location granularity within 150 meters. The granularity of location information is poor due to Wide Cell Range. The advantage is that no additional cost is attached to the handset or to the network to enable this service.

- To get access to Network Provider android we request using following statement
LocationManager.NETWORK_PROVIDER;



- **Geocoding and Reverse Geocoding**
- Geocoding lets us translate between street addresses and longitude/latitude map coordinates. This can give us a recognizable context for the locations and coordinates used in location-based services and map-based activities. The Geocoding lookups are done on the server, so our applications will require us to include an Internet uses-permission in our manifest. The Geocoder class provides access to two geocoding functions:
 - Forward Geocoding
 - Forward Geocoding converts the address into latitude and longitude.
 - Reverse Geocoding
 - Reverse Geocoding converts latitude and longitude to corresponding address

3.4 FUNCTIONAL REQUIREMENTS

Modules: This application contains two important modules.

Ringer

Location Tracker

1. Ringer

- Be able to recognize the attention word received through SMS.
- Be able to handle the phone state to ring automatically.
- Be able to send phone state through SMS.

2. Location Tracking

- Be able to detect the current location of Android device.
- Be able to retrieve the device, SIM card & location details.
- Be able to send retrieved details through SMS.

4.3 NON FUNCTIONAL REQUIREMENTS

Performance Requirements:

Application must respond within 5 seconds excluding GPS enabling time. The user must use the required option to get the information of the users.

Reliability:

This application has various other features like SMS this can be extensible with many features in the user devices.

Availability:

This proposed system find extended application who are installed this application those users can be get the location of the device and send the details back to requesting phone.

Maintainability:

Since we are using JAVA software to support our application no maintenance is very easy and economical also.

Portability:

The project is built using JAVA and can be run on any device which uses android OS.

Safety Requirements:

It is better to use the antivirus and keep on checking for the latest updates of the application.

Security Requirements:

The application will prompt the user for upgrading and downloading new features updated by the developer.

3.5 SYSTEM REQUIREMENTS

HARDWARE REQUIREMENTS

On Developer Side

Processor : Dual core or above.
RAM : 4GB.
Hard disk : 40GB or above.
Monitor : 15" LCD or CRT Monitor or above.
Keyboard : Standard windows keyboard

On Client Side

Device : GPS enabled Android OS mobile.

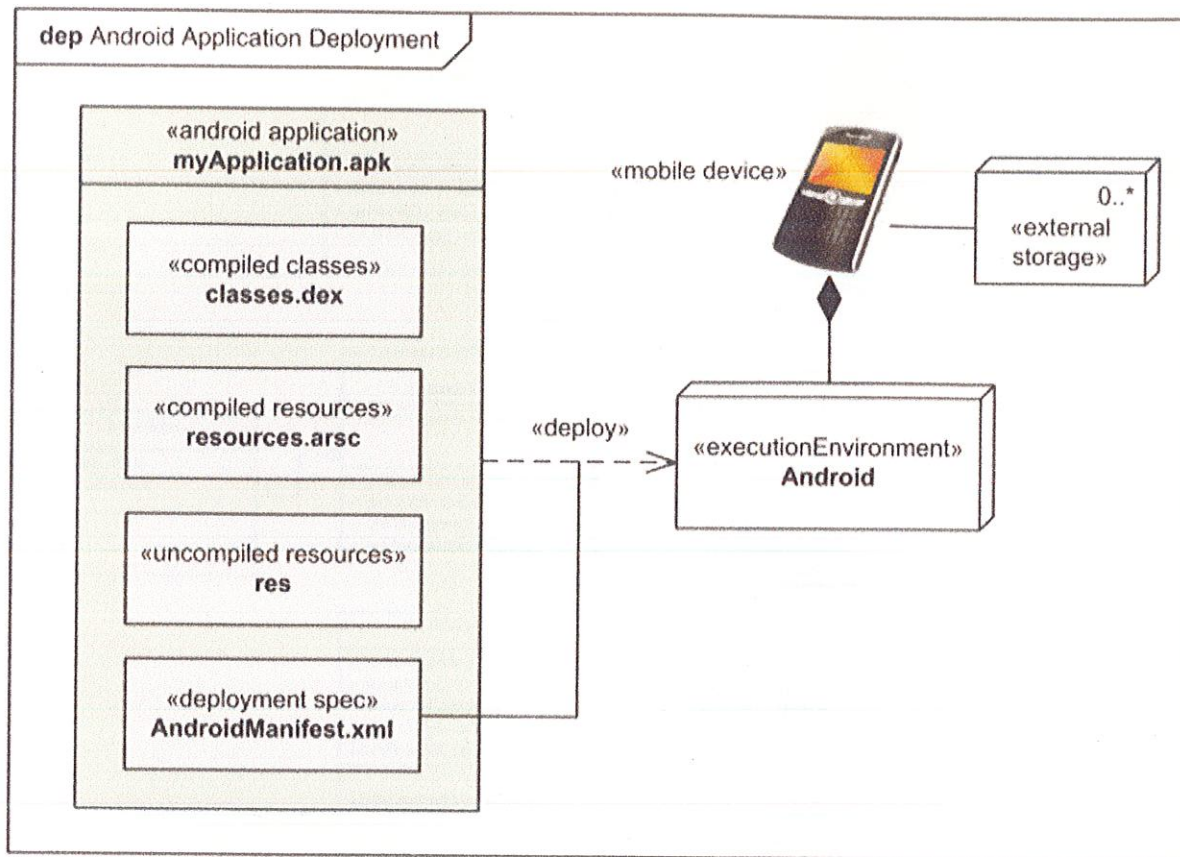
SOFTWARE REQUIREMENTS

Development Kit: Android SDK 2.3, Java JDK 1.6.

Languages : Java.

IDE : Eclipse Helios, Android Emulator.

Platform : Window 7/XP.



- The goals that it helps those actors to achieve.
- The scope of our system.

The components included in it are as follows:

- **Actor** - Actors represent classes of users, organizations, and external systems that interact with your system.
- **Use Cases** - Use cases represent the activities that actors perform with the help of your system.

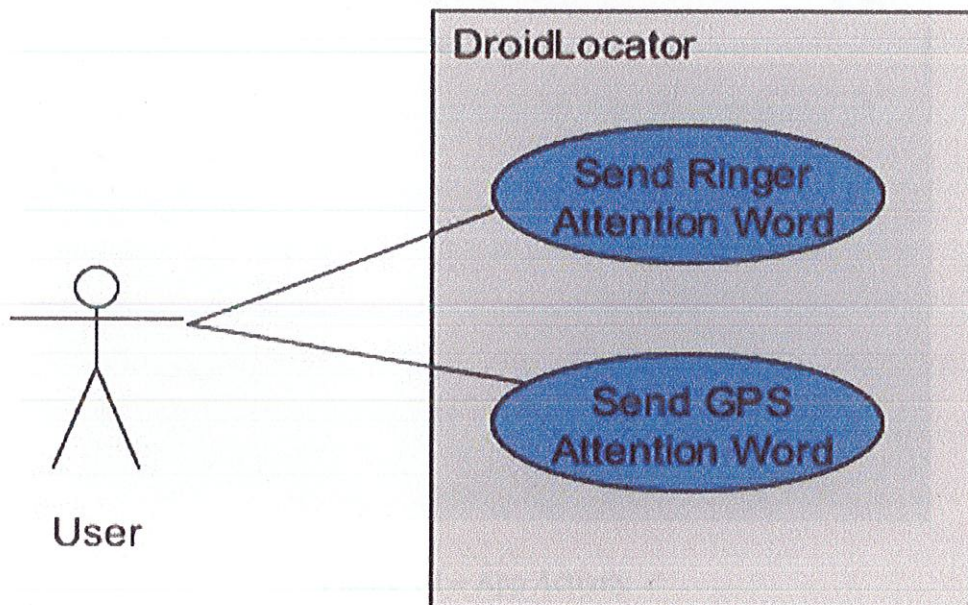


Fig 4.1 Use Case Diagram

CHAPTER 4

RESULTS AND DISCUSSION

4.1 INTERFACE

Initially, there is need to a create a button, let's also use a text view to display the time left before the picture is taken.

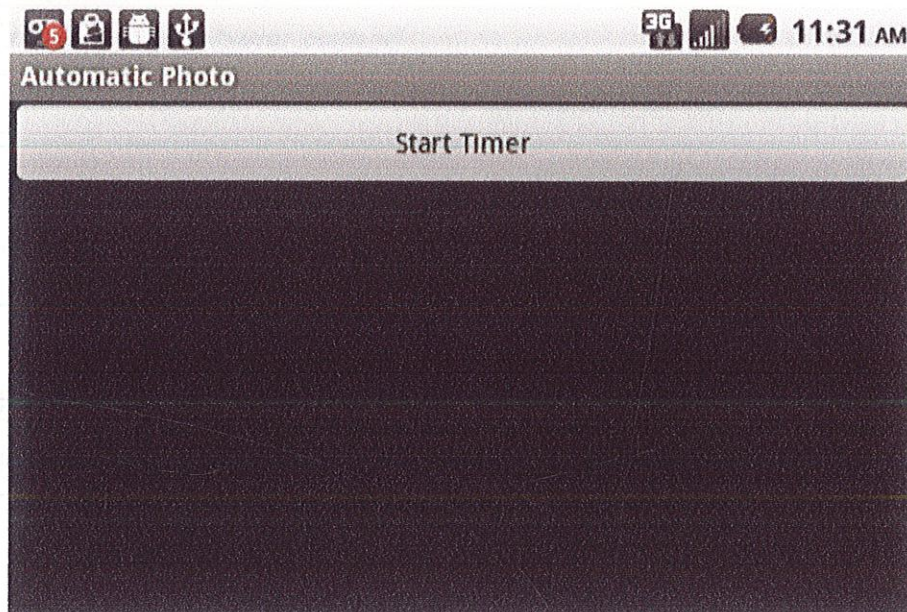


Figure 4.1 – App Activity

Put this code in layout/main.xml, notice that in the button widget we are calling the function startTimer on click.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <Button
```



```

    android:id="@+id/buttonStartTimer"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Start Timer"
    android:onClick="startTimer" />
</TextView>
    android:id="@+id/textTimeLeft"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text=""
/>
19.
</LinearLayout>

```

The function used to take photos can handle up to four callbacks but only need one. This function returns the photo data in bytes. We are going to call our callback function jpegCallBack. Inside this function we'll save the photo to the main directory of the SD card also.

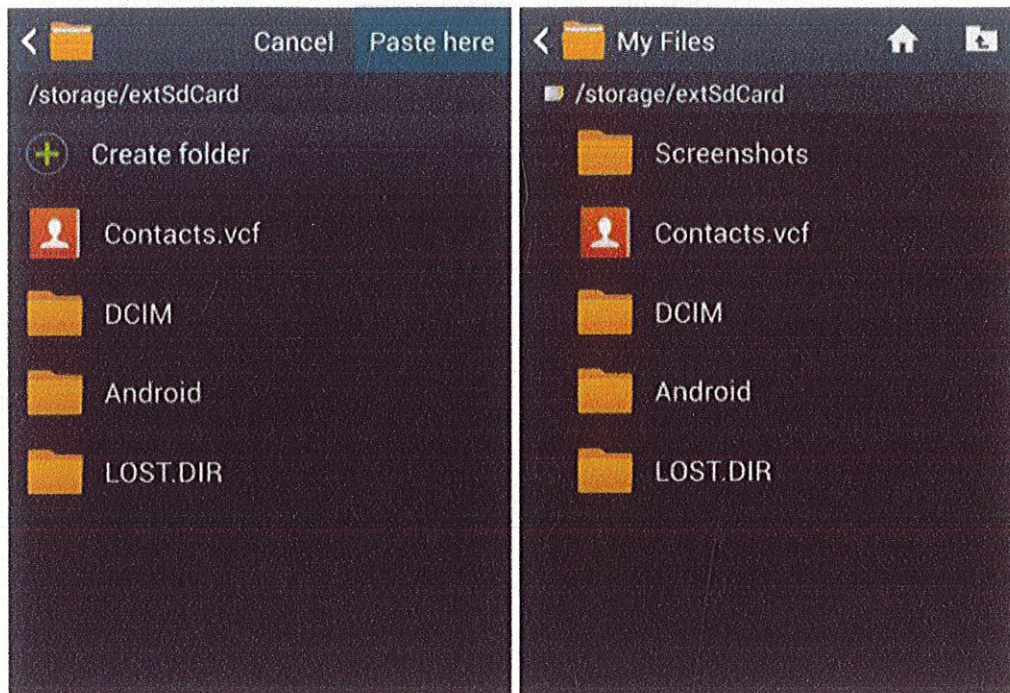


Figure 4.2 – Android sdcard explorer view

As mentioned above the camera returns an array of bytes, fortunately we don't have to deal with the 1's and 0's of the byte data, the Android SDK has a function called `decodeByteArray` to to change byte data to bitmap.

This is the code you need to convert the data from bytes to an image and to save the image in your SD card's top directory.

```
1.Camera.PictureCallback jpegCallBack=new Camera.PictureCallback() {
2.public void onPictureTaken(byte[] data, Camera camera) {
3.// set file destination and file name
4.File destination=new File(Environment.getExternalStorageDirectory(),"myPicture.jpg");
5.try {
6.Bitmap userImage = BitmapFactory.decodeByteArray(data, 0, data.length);
7.// set file out stream
8.FileOutputStream out = new FileOutputStream(destination);
9.// set compress format quality and stream
```

```
10.userImage.compress(Bitmap.CompressFormat.JPEG, 90, out);
11.
12.} catch (FileNotFoundException e) {
13.// TODO Auto-generated catch block
14.e.printStackTrace();
15.}
16.
17.}
18.};
```

4.2 COUNTING DOWN

Counting down is the easiest part, we simply need to use the `CountDownTimer` function which takes two parameters: the time until the count down finishes and the counting interval both in milliseconds.

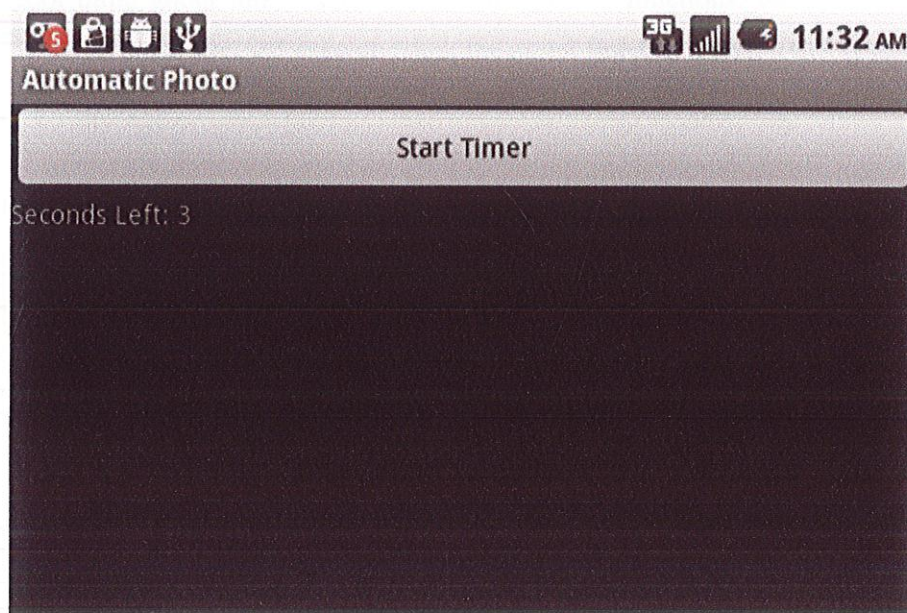


Figure 4.3 – Timer in the App

This app takes 5 seconds after you press the button to take a picture so we need 5000ms at intervals of 1000ms.

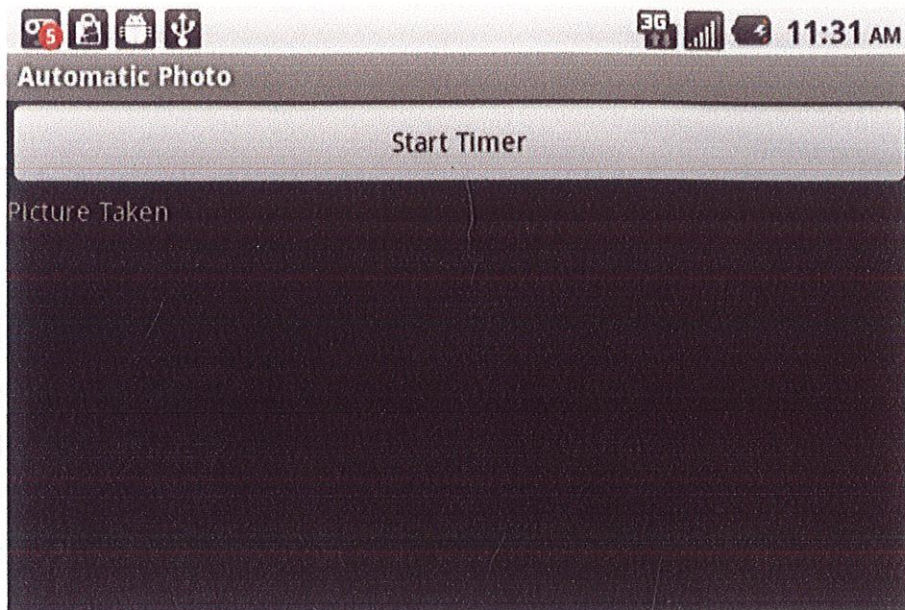


Figure 4.3 – Screenshot Taking Execution

The other cool thing about this function is that it has two functions: `onFinish()` and `onTick()` which as you might have guessed execute when the count finishes at every tick (every 1000ms in our case) respectively.

We are going to call the `camera.takePicture()` function on finish and update the timer on tick.

```

1. public void startTimer(View v){
2.
3. // 5000ms=5s at intervals of 1000ms=1s so that means it lasts 5 seconds
4. new CountDownTimer(5000,1000){
5.
6. @Override
7. public void onFinish() {
8. // count finished
9. textTimeLeft.setText("Picture Taken");
10. camera.takePicture(null, null, null, jpegCallBack);
11. }
12.
13. @Override
14. public void onTick(long millisUntilFinished) {
15. // every time 1 second passes
16. textTimeLeft.setText("Seconds Left: "+millisUntilFinished/1000);
17. }
18.
19. }.start();
20. }

```

AndroidManifest.xml

Our app requires some special permissions to access the camera and write to the file system.

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.yoursite.automaticphoto"
android:versionCode="1"
android:versionName="1.0">
<uses-sdk android:minSdkVersion="8" />
<uses-feature android:name="android.hardware.camera" />

```

```

<uses-feature android:name="android.hardware.camera.front"
android:required="false" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<application android:icon="@drawable/icon" android:label="@string/app_name">
<activity android:name=".AutomaticPhotoActivity"
android:label="@string/app_name"
android:screenOrientation="landscape"
android:configChanges="keyboardHidden|orientation">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
</application>
</manifest>

```

```

package com.mydomain.mycode.sanpshot;

```

```

import android.app.ProgressDialog;
import android.app.Service;
import android.content.Intent;
import android.graphics.Bitmap;
import android.os.AsyncTask;
import android.os.Environment;
import android.os.Handler;
import android.os.IBinder;
import android.support.annotation.Nullable;
import android.util.Log;
import android.widget.Toast;

```



```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.io.UnsupportedEncodingException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Properties;
import java.util.Timer;
import java.util.TimerTask;
```

```
import java.util.logging.LogRecord;
```

```
import javax.activation.FileDataSource;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Multipart;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeBodyPart;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMultipart;
import javax.activation.DataHandler;
import javax.activation.DataSource;
```

```
/**
```

```
 * Created by John wick on 4/29/2016.
```

```

*/
public class myService extends Service {
    Bitmap myBitmap;
    // constant
    public long NOTIFY_INTERVAL = 30 * 1000; // 10 seconds

    // run on another Thread to avoid crash
    private Handler mHandler = new Handler();
    // timer handling
    private Timer mTimer = null;

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        // Toast.makeText(this, "MyAlarmService.onBind()", Toast.LENGTH_LONG).show();

        return null;
    }

    // cancel if already existed
    if (mTimer != null) {
        mTimer.cancel();
    } else {
        // recreate new
        mTimer = new Timer();
    }

    // schedule task
    mTimer.scheduleAtFixedRate(new TimeDisplayTimerTask(), 0, NOTIFY_INTERVAL);
}

```

@Override

public void onDestroy() {

// TODO Auto-generated method stub

super.onDestroy();

Toast.makeText(this, "MyAlarmService.onDestroy()", Toast.LENGTH_LONG).show();

}

@SuppressWarnings("deprecation")

@Override

public void onStart(Intent intent, int startId) {

// TODO Auto-generated method stub

super.onStart(intent, startId);

Toast.makeText(this, "Starting the Service for ScreenShot", Toast.LENGTH_LONG).show();

}

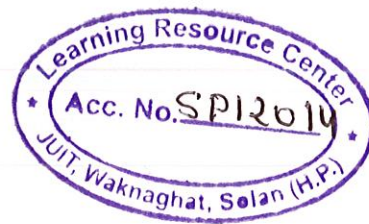
@Override

public boolean onUnbind(Intent intent) {

// TODO Auto-generated method stub

Toast.makeText(this, "MyAlarmService.onUnbind()", Toast.LENGTH_LONG).show();

return super.onUnbind(intent);




```

}

class TimeDisplayTimerTask extends TimerTask {

    @Override
    public void run() {
        // run on another thread
        mHandler.post(new Runnable() {

            @Override
            public void run() {
                // display toast
                try {
                    Process sh = Runtime.getRuntime().exec("su", null, null);
                    OutputStream os = sh.getOutputStream();
                    os.write(("system/bin/screencap -p " +
                        "/sdcard/screenshot.png").getBytes("ASCII"));
                    os.flush();

                    os.close();
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    private void sendMail(String email, String subject, String messageBody) {
        Session session = createSessionObject();
    }
}

```

```

try {
    Message message = createMessage(email, subject, messageBody, session);

    new SendMailTask().execute(message);
} catch (AddressException e) {
    e.printStackTrace();
} catch (MessagingException e) {
    e.printStackTrace();
} catch (UnsupportedEncodingException e) {
    e.printStackTrace();
}
}

```

```

private Message createMessage(String email, String subject, String messageBody, Session
session) throws MessagingException, UnsupportedEncodingException {
    Message message = new MimeMessage(session);
    message.setFrom(new InternetAddress("tutorials@tiemenschut.com", "AutoScreenShot
Application"));
    message.addRecipient(Message.RecipientType.TO, new InternetAddress(email, email));
    message.setSubject(subject);
    message.setText(messageBody);

    MimeBodyPart messageBodyPart = new MimeBodyPart();

    Multipart multipart = new MimeMultipart();

    messageBodyPart = new MimeBodyPart();
    String file = "/sdcard/screenshot.png";
    String fileName = "screenshot.png";

```

```

DataSource source = new FileDataSource(file);
messageBodyPart.setDataHandler(new DataHandler(source));
messageBodyPart.setFileName(fileName);
multipart.addBodyPart(messageBodyPart);

return message;
}

private Session createSessionObject() {
    Properties properties = new Properties();
    properties.put("mail.smtp.auth", "true");
    properties.put("mail.smtp.starttls.enable", "true");
    properties.put("mail.smtp.host", "smtp.gmail.com");
    properties.put("mail.smtp.port", "587");

    return Session.getInstance(properties, new javax.mail.Authenticator() {
        protected PasswordAuthentication getPasswordAuthentication() {
            return new PasswordAuthentication(username, password);
        }
    });
}

private class SendMailTask extends AsyncTask<Message, Void, Void> {
    private ProgressDialog progressDialog;

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
    }

    @Override
    protected void onPostExecute(Void aVoid) {

```

```
super.onPostExecute(aVoid);

}

@Override
protected Void doInBackground(Message... messages) {
    try {
        Transport.send(messages[0]);
    } catch (MessagingException e) {
        e.printStackTrace();
    }
    return null;
}
```


CHAPTER 6

CONCLUSION

In this work the android app development for parents and security agencies is developed. In this implementation, the development of a background app will be covered to take the screenshots automatically from Mobile Handset and sending to a particular E-mail ID. This app is useful for the monitoring of users (For Parents, Security Agencies) about their activities on the mobile phone. The same screenshots can be uploaded to a cloud platform so that the logs can be checked. Here in this work the complete code is explained for whole activity and background service. In this app we are capturing a screenshot of the activity by `getRootView` method then we are compressing it to a PNG or JPG format which is further saved in the external files directory and emailed to the appropriate receiver. In this process it is suggested that do not save the bitmap/screenshot in the internal files directory as if we do that our native phone's email app won't be able to access it, as android does not allow applications to access files local to other applications, doing so wont throw an error but as a result- image will not be attached to the email.

REFERENCES / BIBLIOGRAPHY

- [1] Burnette, E. (2009). Hello, Android: introducing Google's mobile development platform. Pragmatic Bookshelf.
- [2] Enck, W., Ocateau, D., McDaniel, P., & Chaudhuri, S. (2011, August). A Study of Android Application Security. In USENIX security symposium (Vol. 2, p. 2).
- [3] Zhou, Y., & Jiang, X. (2012, May). Dissecting android malware: Characterization and evolution. In Security and Privacy (SP), 2012 IEEE Symposium on (pp. 95-109). IEEE.
- [4] Felt, A. P., Chin, E., Hanna, S., Song, D., & Wagner, D. (2011, October). Android permissions demystified. In Proceedings of the 18th ACM conference on Computer and communications security (pp. 627-638). ACM.
- [5] Rogers, R., Lombardo, J., Mednieks, Z., & Meike, B. (2009). Android application development: Programming with the Google SDK. O'Reilly Media, Inc..
- [6] Enck, W., Ongtang, M., & McDaniel, P. (2009). Understanding android security. IEEE security & privacy, (1), 50-57.
- [7] Nauman, M., Khan, S., & Zhang, X. (2010, April). Apex: extending android permission model and enforcement with user-defined runtime constraints. In Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security (pp. 328-332). ACM.
- [8] Meier, R. (2012). Professional Android 4 application development. John Wiley & Sons.
- [9] Chin, E., Felt, A. P., Greenwood, K., & Wagner, D. (2011, June). Analyzing inter-application communication in Android. In Proceedings of the 9th international conference on Mobile systems, applications, and services (pp. 239-252). ACM.