**Chapter 1: INTRODUCTION**

**1.1 Problem definition**

User/owner trail identification using Android Forensics.

**1.2 Background for the project**

Many a times it is observed that when a stray mobile device is seized, the common man does not have expertise to identify the owner of the device. Hence there is a need of an easy mechanism for a common man to get the traces of the owner/user of the mobile device. In order to identify the owner, the most important prerequisite is that the end user must be well aware of the technology and the way of using software. After having achieved this, our purpose for project making is solved.

**1.3 Domain**

Android forensics: Conducting forensic data analysis. The tests and analysis are performed with the aim of determining what data and information can be found on the devices internal memory as well as external memory, images, call logs etc. e.g. chat messaging logs and history send & received image or video files etc. The experiments and results show that heavy amount of potential evidences and valuable data can be found on Android phones by forensic investigators.

**1.4 Motivation**

When a handheld device is recovered from a location, many a time it is challenge for police or any other person to identify the owner of this handheld device. A guided recovery software can be used to recover vital information from handheld device. This includes identification parameters, browser data, contacts and data (if any) from various installed applications. Guided software can be developed for one kind of OS; we are developing for a specific android version by looking at the standard sources of digital evidences. Some of these sources are digital evidences that come in many forms like Application Logs, Application meta-data, Database Contents, Transaction Records, Call logs, Message Records, Chat

Application Logs, Footages, Images/Pictures. The list is not limited to above.

**1.5 Android data analysation tool as a solution**

This is a forensic data extraction and analysis tool for version 4.0 of the Android platform. The tool incorporates multiple scripts written in python. It can automatically dump predefined SQLite database files from Android devices and extract the contents stored in the dumped database.

In the first step ,this tool establishes a connection to an Android device via Android debugging bridge (adb) , dumps the predefined SQLite database files from the phone and stores them on investigators machine (dump module). All of the subsequent steps are performed on the copies of the database files. In second step, the contents of dumped

database file copies are analyzed and then report is generated.

**Chapter 2: LITERATURE SURVEY**

**2.1 Introduction**

Today’s most vulnerable device in use is the mobile device (Smartphone) whose data capture has always been a challenging thing. There are number of Mobile Operating Systems present in the market. Among these Mobile OS, Android, iOS and RIM are more popular than others. Android is the most widely used Mobile OS present in the market. It is quite obvious that the widely used platform is likely to be targeted more. And as Android smart phones are gaining popularity they are gaining the attention of hackers too. It is always a challenge for forensic examiners to discover the evidences from the Android devices. Android has a different and newer file system, directory structure, runtime environment, kernel and libraries which make Android more complex to forensic examiner. Still there are some investigators/examiners who work towards maintaining integrity of the vulnerable device. Here comes Android forensics in picture. Android forensics is a branch of [digital forensics r](https://en.wikipedia.org/wiki/Digital_forensics)elating to recovery of [digital evidence o](https://en.wikipedia.org/wiki/Digital_evidence)r data from an android device under [forensically sound](https://en.wikipedia.org/wiki/Forensic) conditions

**2.2 Types of evidences**

The following are types of potential digital evidence that may be found in the Smartphone devices:

* Call history

The call history provides an insight to the call activity of the owner before the acquisition of the Smartphone device. The investigator can see in-coming, out-going and missed calls including their time and durations. This can help the forensics investigator to draw indirect conclusion about the suspected activities.

* Contact list

The contact list not only provides contact names and their numbers either home, mobile and work but also many other types of information such as contact title, company, address and emails. Also, some Smartphone devices store a picture of the contact in the contact list. The information that is stored in the contact list provides the investigator with the social and work relations of the owner of the Smartphone

device. Beside this, many people store different types of account information and password. Example bank accounts, pin code etc

* Text messages/ Emails

Contrary to the call history and contact list which provide indirect information, text messages and emails give explicit information that can be used as evidence in the court. This is because they contain the exact text intended to or sent by the owner of the Smartphone device.

* Media (pictures, videos, audio)

Media files such as pictures and videos can be used as potential digital evidence in the court. Many Smartphone devices such as iPhones embed the GPS co-ordinates of the location into the metadata called Exchangeable File Format (Exif) of the resulting image file (Valli & Hannay, 2010). Not only are the GPS co-ordinates stored but also valuable information for the investigator such as the date and the time of capturing. This provides the investigator with more insight of the activities of the owner of the Smartphone.

* Browsing history/internet search

The browsing history and internet searches in the Smartphone device give the investigator a picture of the internet activities of the owner. The investigator will discover the types of web sites that the owner has visited. Also, some Smartphone devices give the owner the ability to save their favorite web sites.

* Chat logs

There are several chat applications that can be installed in the Smartphone device such as Windows Live Messenger, Google Talk and BlackBerry Messenger. Users of these applications usually choose to save the chat logs. The chat logs can be used as digital evidence in the court as to what the owner said.

* Social network accounts

Most Social networks are available on Smartphones, including the most famous of all, Facebook. In this type of account the investigator can find pictures and notes that

were published by the owner. Also, they can discover the owner's friends and the groups that they belong to.

* Calendar\ notes

The calendar gives a picture of the previous, current and future planned activities of the owner of the Smartphone. The calendar can be used to associate the owner of the Smartphone to specific locations and times in order to look for possible witnesses. The owner of the Smartphone may also have saved notes that have valuable information that can be presented as evidence in the court.

* Connections (mobile network, Wi-Fi, Bluetooth)

These will give the investigator an overview of the networking activities that were performed by the owner‘s Smartphone device. The mobile network will give a picture of which country or region the owner has roamed in. Wi-Fi will give a picture of which Local Area Network (LAN) the Smartphone connected to. Bluetooth will give the forensic investigator information about the nicknames of the devices that were connected with owners Smartphone using Bluetooth connection.

* Maps (locations, directions help, favorites)

This will provide the investigator with a geographical view of the owner‘s movements which can be used as potential evidence in court.

**2.3 Survey of existing systems**

Extraction tools can be hardware or software depending on how the data is extracted from mobile device. There are a number of extraction tools available today and newer tools are emerging with some innovative ideas. Some of them are listed below

1. AFlogical

AFlogical OSE is an open source tool to extract the data. This is lightweight software purely used in the command line. This tool utilize Android adb feature to communicate with a computer

2. Oxygen Forensic

Oxygen Forensic is one of the leading tools in the mobile forensic filed with a wide range of phone support. Oxygen extracts most of the information in an efficient way.

this tool has a well defined reporting system so that examiner can read and verify minute details of the evidence collected.

3. MOBILedit Forensic

MOBILedit Forensic tool allows examiners to acquire logicallly, search and examine the mobile phone devices. This tool uses multiple connectivity mechanism than other similar tools especially the wireless connectivity. The software is well enough to acquire the phone system information and other contacts and messages list.

4. Encase-

It’s a tool which produces extensive reports on your findings while maintaining the integrity of your evidence. No other solution offers the same level of functionality, flexibility, and has the track record of court-acceptances Encase Forensic.

5. FTK Imager-

The FTK Imager is a simple but concise tool. It saves an image of a hard disk in one file or in segments that may be later on reconstructed. It calculates MD5 hash values and confirms the integrity of the data before closing the files. The result is an image file(s) that can be saved in several formats, including DD raw

**2.4 Limitations of existing systems**

Most of these tools are expensive.Some tools like Encase require extensive training which makes it difficult to use by layman. Encase have many features, but these features often carry a price of complexity and also the GUI isn’t intuitive.

These tools cannot store log files for actions taken by examiners during analysis actions. Extensive search customization that can be a confusing disadvantage to inexperienced examiners.

**2.5 Brief overview of papers referred**

**1. Android forensics simplifying cell phone examination**

Jeff Lessard, Gary C. Keseler - September 2010

One of the first works conducted in this area was done by Lessard and Kessler (Lessard& Kessler, 2010). The authors investigated an Android Smartphone by acquiring a logical and physical image of the device. They also used Cellebrite, a mobile forensic tool, to acquire information from the device and perform a comparison between these methods. The result of the analysis showed that the logical examination had resulted in the least fragmented files and easily viewable data. However, an issue with this research is the use of the dd command to obtain a physical image of the device.

**2. Forensic analysis of the android file system YAFFS2**

Timothy Vidas et al. (Vidas, Zhang, &Christin, 2011)

This paper proposed a new methodology for forensic data collection for Android devices. The methodology relies heavily on the recovery partition of the device. In this data was extracted in different formats by using a variety of software processes, such as SuperOneClick, dd, xRecovery, NANDdump, Yaffs2utils and Android Debug Bridge. Analysis of the extracts was then undertaken to determine the type of data available from the different extraction methods

**3. Forensic analysis of smart phones: The android data extractor lite (ADEL)**

Felix Freiling, Michael Spreitzenbarth, Sven Schmitt -2011

Felix Freiling steps up a quite similar environment like we do by explicitly dumping the SQLite databases from the device and display the extracted data as the outcome of his work. This system only considers telephone and SIM-card information (e. g. IMSI and serial number) telephone book and call lists, calendar entries, SMS messages.

**4. Guidelines for the digital forensic processing of smart phones**

Khawla Abdulla Alghafli1, Andrew Jones1, Thomas Anthony Martin1-2011

In this paper various types of crime and their associated digital evidence are discussed. The digital forensics process of the Smartphone devices is discussed and, this paper also contains recommended guidelines and procedures for how to perform the phases of the digital forensics process on Smartphone devices.

**Chapter 3: SOFTWARE REQUIRMENT SPECIFICATION (SRS)**

**3.1 Introduction**

It’s a description of a software system to be developed. It lays out functional and non

functional requirements and may include a set of use cases.

**3.2 Purpose of the document**

The purpose of this document is to present a detailed description of the problem definition i.e. identifying user trails using Android forensics. It will explain the purpose and features of the system which will help us identify the same as mentioned above. It will describe the functioning of the system to be developed, constraints under which it must operate, and its reaction (the actual output) towards external stimuli (inputs provided).

**3.3 Scope of development project**

The proposed project considers only android mobile devices. The data analysis is limited to collection of information from calls, messages, contacts, pictures, videos, updates, chat logs, documents and various applications. The usage of some of the existing mobile forensic tools in this project is permitted.

The development platform is Android operating system ecosystem.

**3.4 Strength and weakness**

**Strengths:**

* Product does not require online access
* Programmed to identify the device besides analyzing the same
* Filters put extraneous data, generates brief summary in the report.

**Weakness:**

* Considers only android devices (specific family).
* Cannot work on non-rooted phones.
* Process once started cannot be reversed , has to be re-started all over again

**3.5 General Description**

**3.5.1 User personas and Characteristics**

Users of the software should be able to retrieve information about owner/user of the mobile device. The software will be used by investigators, with basic knowledge about computers and mobile devices. Investigator should be able to do the following things:

Connect a device and give confirmation about it

Should be able to view all the information needed to identify the owner/user of device.

**3.5.2 Product perspective:**

It provides the end user with a mechanism to retrieve the relevant information

**3.5.3 Overview of functional requirements**

Extraction: Here the input will be the data extracted from the internal and external memory of the android device. eg. chat messaging logs, history of send and received emails, messages, images, calls etc .

After obtaining the data following operations will be performed on it:

1. Querying: User trails can be identified querying the obtained database

2. Report generation: output of queries will be seen in the report generated

|  |  |
| --- | --- |
| Function | Extraction of databases |
| Pre-condition | 1.phone is connected properly and rooted  2.phone is unlocked |
| Steps | Click on Ok button to initialize extraction |
| Post condition | All required databases are extracted successfully |
| Alternative Flows | 1. If phone is not connected properly device will be seen offline in this  case extraction of databases is not possible.  2. If the phone is not rooted access to databases is denied. |

Table 3.1 Functional Requirements 1

|  |  |
| --- | --- |
| Function | Report generation |
| Pre-condition | All databases are extracted successfully |
| Steps | Select the options for which report is supposed to be generated and then  click on OK button |
| Post-condition | Details of selected options are seen in the report generated |
| Alternative Flows | On unsuccessful extraction of database the report will not be generated |

Table 3.2 Functional Requirements 2

**3.5.4 Overview of Data Requirements**

File system acquisition- File system extraction is useful for understanding the file structure. Databases - databases extracted from the internal and external memory of the android device

gives data of. chat messaging logs, history of send and received emails, messages, images, calls etc .

**3.5.5 Operating Environment**

The software needs following software and applications:-

* The software being developed operates under any Linux based operating system which has adb and python.
* The mobile device used will be Android OS, v4.0 (Ice Cream Sandwich).
* The databases, on which we will be operating, are SQLite databases.

**3.5.6 General constraints, assumptions, dependencies, guidelines**

General constraints-

The proposed project considers only the Android mobile devices and Linux based operating system.

Assumptions-

The end user is a non technological person

The mobile device is in unlocked mode.

The system works satisfactorily under the environment of given constraint.

Dependencies-

The system would give accurate results in the presence of accurate data retrieval making it data dependent.

As per the given constraints it is platform dependent as well.

**3.6 Specific Requirements**

**3.6.1 External Interface Requirements**

USB cable

**3.6.2 Performance Requirements**

Performance of the software depends upon the data it gets from the device to analyze.

The data, in this case data, shouldn’t be tampered. If the data is intact chances of getting proper results are high. The time taken by the software to deliver the results depends on the data collected from the device.

The data which is acquired from the device is another factor. If the data acquired is huge it may slow down the process.The speed of software depends on USB cable which is used for connecting mobile device with the computer

**3.6.3 Other Requirements**

Android OS v4.0 (Ice Cream Sandwich)

Linux based operating system

Android debug Bridge

SQLite

**3.7 Non-functional requirements**

Usability: The use of this guided software is intended to be as simple as possible to allow its use by both qualified persons and non-experts. At best, the analysis of the mobile phone is conducted in an autonomous way so that the user does not receive any notice of internal processes. Moreover, the report module creates a detailed report in a readable form.

Availability: The system can work 24x7 without any glitches or downtime.

Maintainability: The system is a onetime setup system, which does not require any further additional installations or hardware components.

Expandability: Allowed required modifications at appropriate locations to be made without undesirable effects.

**Chapter 4: ARCHITECTURE AND DESIGN**

**4.1 Mathematical model**

S={s, e, X, Y, DD, NDD, Fmain, Ffriend , CPUcorecount , MEMshared, Success, Failure}

s=start state e=end state X=set of inputs Y=set of outputs

DD=deterministic data

NDD =non deterministic data

Fmain = Main function Ffriend = Friend function CPUcorecount = CPU corecount MEMshared = Memory shared

Success = Required outcome generated

Failure = Required outcome not generated

S' = {s, e, X, Y, DD, Fmain, Ffriend , Success, Failure} X= {acquired\_databases}

acquired\_databases = Call logs, Message Records, Chat Application Logs, Images/Pictures, browsing history, Gmail.

Y={user\_identification} Fme = {Fk1 ,Fk2 ,Fk3 ,Fk4}

FK1 =database\_acquistion()

acquire all databases from mobile handset € X

Fk2 = database\_querying()

query the obtained database to get user trails

Fk3 =database\_encryption()

encrypt the database to maintain database integrity

Fk4 =report\_generation()

report contains last jpg files accessed, top users communicated on Whats app , top people contacted (SMS/Whats app/ call logs), frequently visited URL, call logs ,saved URL's and password , message digest of top 5 communicated people on Whats app.

DD={X}

Success = database successfully acquired and user trails identified

Failure= {f1 ,f2 ,f3}

f1 = original data gets corrupted

f2 = alteration/tampering of original data. f3 =report not generated

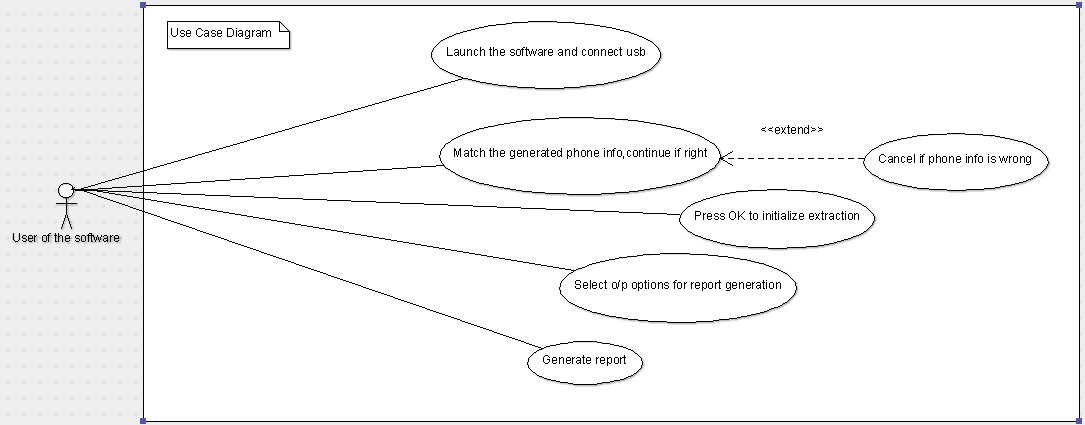


Fig 4.1 Use case diagram

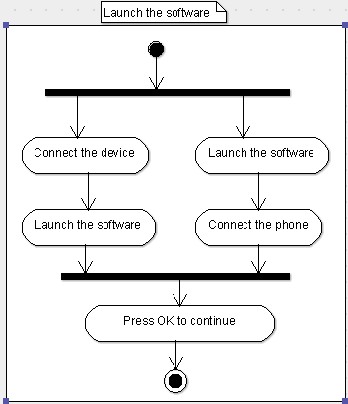


Fig 4.2 Activity diagram to launch the software

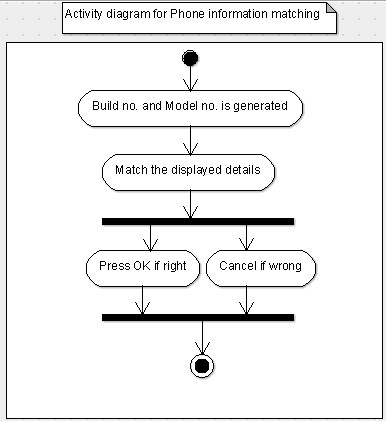


Fig 4.3 Activity diagram for Phone information matching

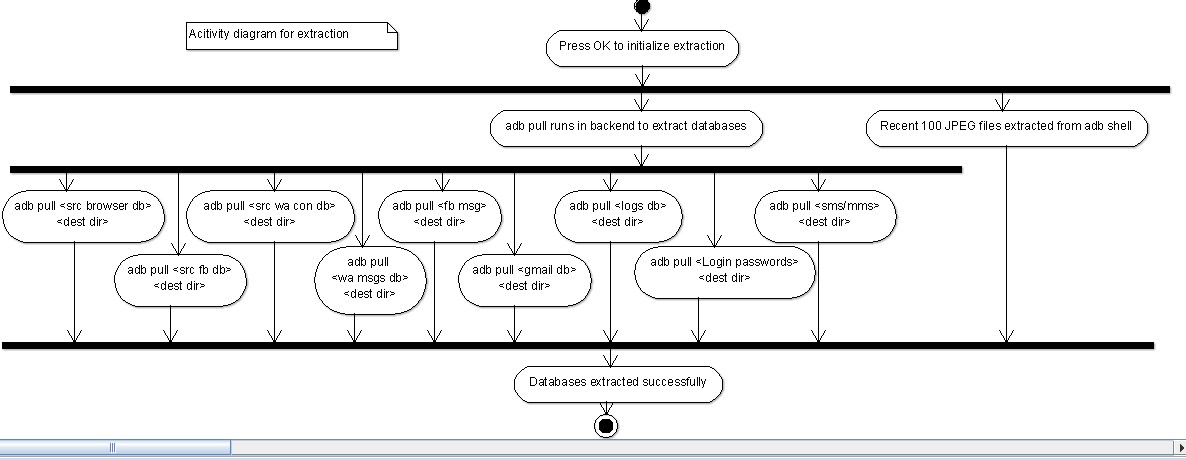


Fig 4.3 Activity diagram for extraction

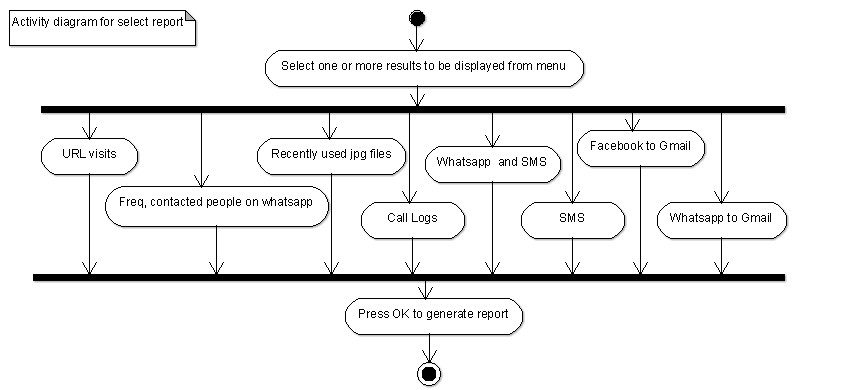


Fig 4.4 Activity diagram for report selection

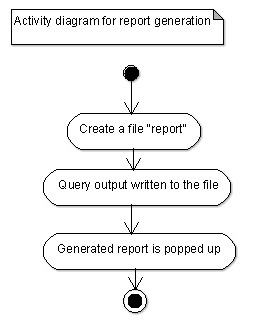


Fig 4.5 Activity diagram for report generation

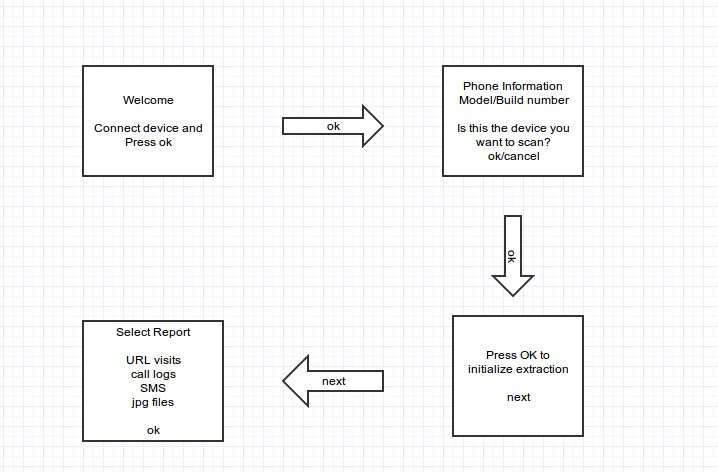


Fig 4.6 Screen design

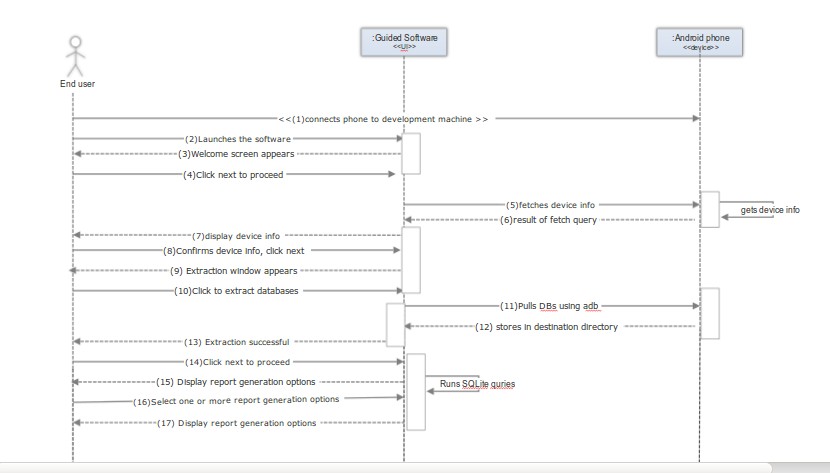


Fig 4.7 Sequence diagram

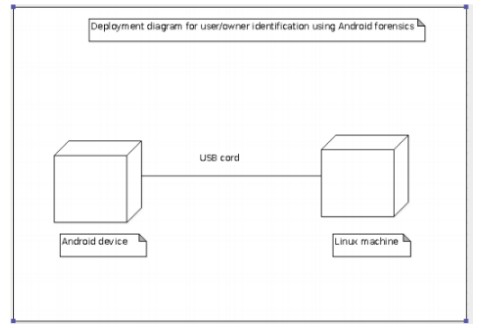


Fig 4.8 Deployment diagram

**4.11 Methodologies**

**4.11.1 Rooting methodology**

Rooting an an android device is equivalent to jail breaking , it means you have linux like super user permissions. Gaining access to the core software of your device makes it highly

vulnerable and can result into bricking the device if not done properly. By unlocking the android subsystem , the operating system , we can install the unapproved apps, replace the stock ROM by a custom ROM, update the OS, speed up the phone by overclocking and increase battery life by updating the kernel. Unlocking your bootloader(which the software at lowest level in phone) renders the device warranty void , even if its is unrooted , once rooted phone is very easy to recognize by manufacturer. The XDA developersis an authentic forum to look up for apks that help in rooting and gather useful information about the device internals.

Steps to root

1. Download and install KingRoot apk
2. USB debugging mode has to be enabled on phone. Tap Settings, Developer Options, then tick the box for "USB debugging." Tap OK to approve the setting change.
3. Run Android Root on PC, then connect phone via its USB sync cable. The device screen may show an "Allow USB debugging?" pop-up. Tick "Always allow from this computer," and tap OK.
4. Click Root and let the utility do it's work. The process can be reversed, run Android Root again, connect phone, then click Remove Root.

**4.11.2 Extraction methodology using ADB:**

When adb is started, it checks if there is an adb server process already running. If there is not, it starts server process. When server starts it binds to local TCP port 5037 and listens for commands sent from adb client, all adb clients use adb clients use port 5037 to communicate to adb server. To query via adb, the flow the commands is as follows

1. adb devices(gives device model and ID)
2. adb root (runs the adbd daemon in root)
3. adb pull<source file path in android sub-system> <destination path in development machine>

Following table gives module names and their corresponding paths to pull from.

|  |  |  |
| --- | --- | --- |
| 1  . | SMS/MMS | /data/data/com.android.providers.telephony/databases/mmsms.db |
| 2  . | Default  Browser | /data/data/com.android.browser/databases/browser 2.db |
| 3  . | Whatsapp  Contacts | /data/data/com.whatsapp/databases/wa.db |
| 4  . | Gmail | /data/data/com.google.android.providers.gmail/databases/mailstore.accoun  t\_name.in.com.db |
| 5  . | Whatsapp  Calls | /data/data/com.whatsapp/databases/msgstore.db |
| 6  . | Facebook  messenger | /data/data/com.facebook.orca/databases/threads\_db2 |

**4.12 UI Screenshots**

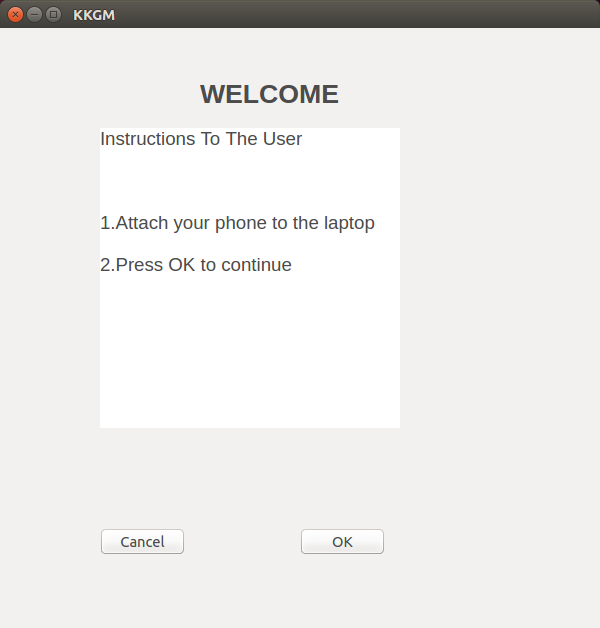


Fig 4. 1 Welcome window

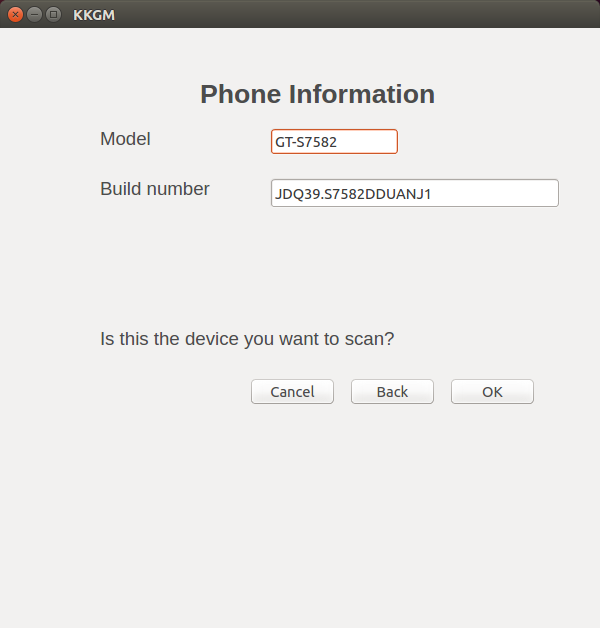


Fig 4. 2 Phone information

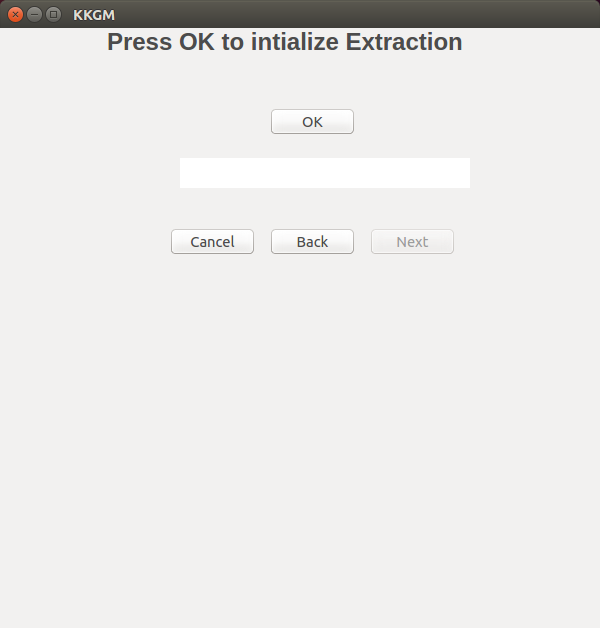


Fig 4. 3 Extraction Window (before extraction)

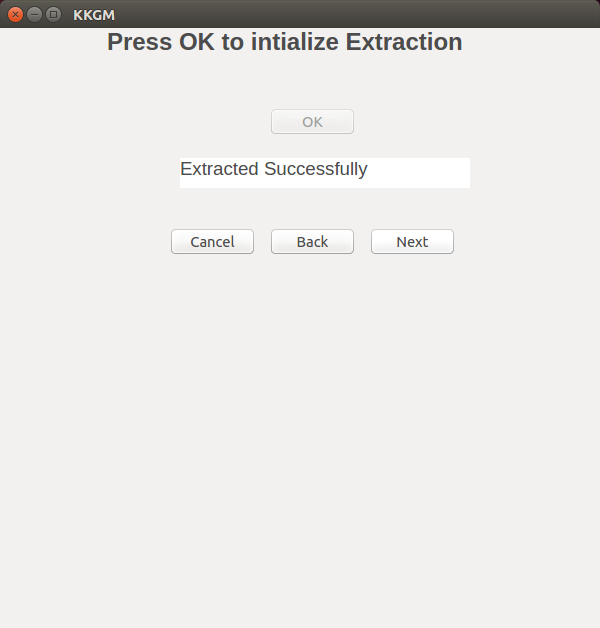


Fig 4.3.1.2 Extraction Window (after extraction)

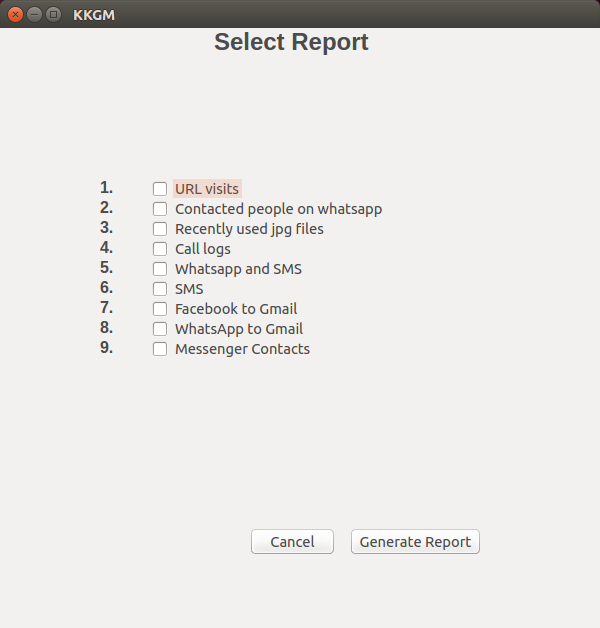


Fig 4.4 Report selection window

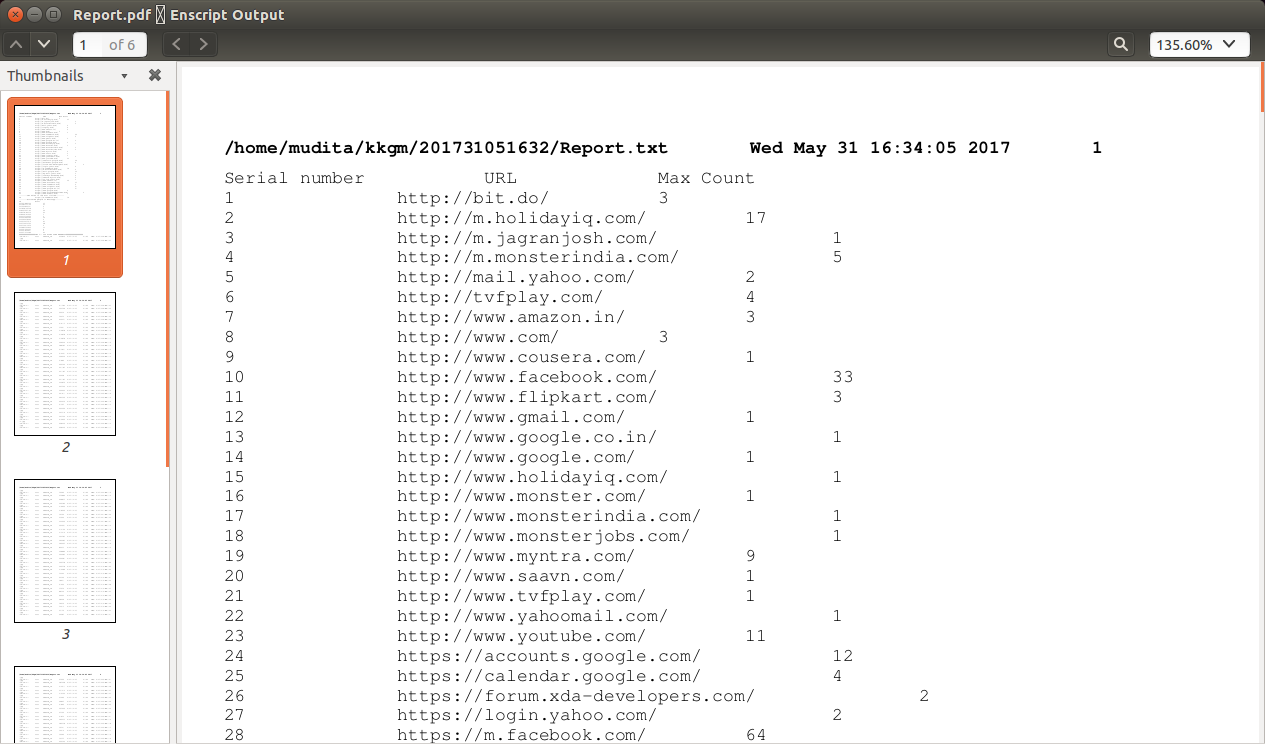


Fig 4.5 Report generated

**Chapter 5: TECHNOLOGY**

**5.1 SQLite**

SQLite (Structured Query Language) Lite is a compact C programming library that resides in a single cross-platform disk file and contains its own relational database management system. SQLite has application programming interfaces (APIs) that enable it to run in the same process with coded applications. SQLite reads and writes directly to disk files and to database entities including tables, indices, triggers and views.

SQLite has a small footprint, facilitating its use in web browsers and in portable devices such as smart phones and tablets. SQLite is accessible through C, PHP Hypertext Pre-processor (PHP), Transmission Control Protocol (TCP), Ruby, Perl and Python.

**5.2 PyQt**

PyQt is one of the two most popular Python bindings for the Qt cross-platform GUI/XML/SQL C++ framework. PyQt developed b[y Riverbank Computing Limited.](http://www.riverbankcomputing.com/) PyQt is a set of Python v2 and v3 bindings fo[r The Qt Company's Qt](http://www.qt.io/) application framework and runs on all platforms supported by Qt including Windows, OS X, Linux, iOS and Android. PyQt5 supports Qt v5. PyQt4 supports Qt v4 and will build against Qt v5. The bindings are implemented as a set of Python modules and contain over 1,000 classes.

**5.3 Python**

Python is an [interpreted](http://whatis.techtarget.com/definition/interpreted-script)[, object-oriented programming](http://searchsoa.techtarget.com/definition/object-oriented-programming) language similar to [PERL,](http://searchenterpriselinux.techtarget.com/definition/Perl) that has gained popularity because of its clear [syntax a](http://searchcio-midmarket.techtarget.com/definition/syntax)nd readability. Python is said to be relatively easy to learn and portable, meaning its statements can be interpreted in a number of [operating systems,](http://searchcio-midmarket.techtarget.com/definition/operating-system) including [UNIX-](http://searchenterpriselinux.techtarget.com/definition/Unix)based systems, [Mac OS](http://whatis.techtarget.com/definition/Mac-OS)[, MS-DOS](http://searchenterprisedesktop.techtarget.com/definition/MS-DOS)[, OS/2,](http://whatis.techtarget.com/definition/OS-2) and various versions of Microsoft [Windows 98.](http://searchenterprisedesktop.techtarget.com/definition/Windows-98) A notable feature of Python is its indenting of source statements to make the code easier to read. Python offers dynamic [data type, r](http://searchsoa.techtarget.com/definition/data-type)eady-made [class, a](http://whatis.techtarget.com/definition/class)nd

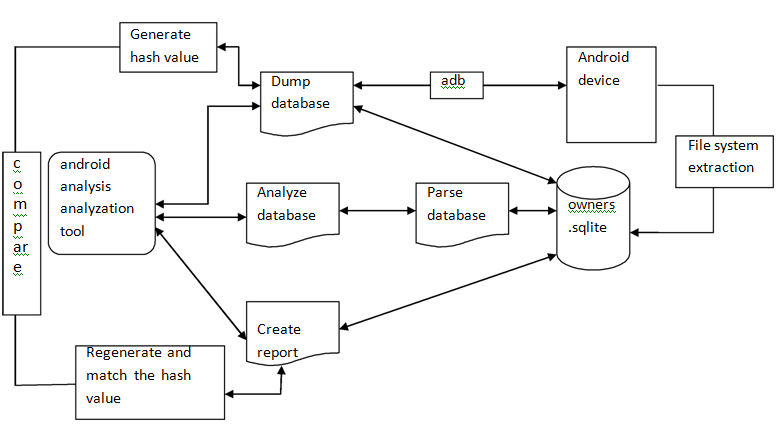
interfaces to many system calls and libraries. It can be extended, using the [C or](http://searchwinit.techtarget.com/definition/C) [C++ la](http://searchsqlserver.techtarget.com/definition/C)nguage.

**Chapter 6: IMPLEMENTATION AND CODING**

**6.1 Modules**

* Extract database and file system of an android device
* Calculate the hash value of obtained databases
* Analyze database and file system
* Parse database in a new database (owners.sqlite)
* generate report
* And recomputed the hash value of databases and compare it with initially calculated values to ensure data integrity

**6.2 Flow diagram**



**6.3 Algorithm used**

The MD5 algorithm is a widely used [hash function p](https://en.wikipedia.org/wiki/Hash_function)roducing a 128-[bit h](https://en.wikipedia.org/wiki/Bit)ash value. It can still be used as a [checksum to](https://en.wikipedia.org/wiki/Checksum) verify [data integrity.](https://en.wikipedia.org/wiki/Data_integrity) The Hashlib package can check data integrity

by making use of hash algorithms. It provides an interface to generate the hash value of the result set returned by a SQLite query. In this project we are using MD5 to calculate the hash value of the database during database extraction and then we are again generating hash values during report generation to verify the integrity of the database.

**6.4 Codes and snippets**

6.4.1 Model and build number display of mobile device

status,output=commands.getstatusoutput("adb shell getprop ro.product.model")

self.edit1.setText(output)

status,output=commands.getstatusoutput("adb shell getprop ro.build.display.id")

6.4.2 Extraction function

def extractButton(self):

self.button1.setEnabled(False)

sqlite\_file2= 'owners.sqlite'

table\_name1='accounts'

column\_1 = 'name'

db\_path1='/data/system/users/0/accounts.db'

conn2= sqlite3.connect(sqlite\_file2)

cursor= conn2.execute('select count(serial\_no) from path\_name')

for row in cursor:

count=row[0]

count=(count+1)

for i in range(1,count+1):

cursor= conn2.execute('SELECT full\_path from path\_name where serial\_no={ino}' .\

format(ino=i))

for row in cursor:

db\_path= row[0]

status,output = commands.getstatusoutput("adb pull '%s' '%s'/kkgm/'%s'/" % (db\_path,path,date\_time))

os.chdir("%s/kkgm/%s" % (path,date\_time))

status,output = commands.getstatusoutput("adb pull '%s' '%s'/kkgm/'%s'/" % (db\_path1,path,date\_time))

conn1=sqlite3.connect('accounts.db')

cursor1=conn1.execute("select {cn\_1} from {tn} where {cn\_1} like '%@%' ".\

format(cn\_1=column\_1,tn=table\_name1))

for row in cursor1:

acc=row[0]

status,output = commands.getstatusoutput("adb pull /data/data/com.google.android.gm/databases/mailstore.'%s'.db '%s'/kkgm/'%s'/mailstore.db" % (acc,path,date\_time))

status,output = commands.getstatusoutput("cp '%s'/owners.sqlite '%s'/kkgm/'%s'/" % (path,path,date\_time))

os.system('adb shell ls -lR | grep "jpg"| sort -r -k 5 | tail -100 | tee jpg17.txt > /dev/null 2>&1')

os.system('sed "s/[[:space:]]\+/,/g" jpg17.txt > topjpg.csv')

self.button2.setEnabled(True)

self.label2.setText("Extracted Successfully")

6.4.3 Url visits function

def url\_visits(self,state):

if self.checkBox1.isChecked()== True:

sqlite\_file1 = 'History'

sqlite\_file2 = 'owners.sqlite' # name of the sqlite database file

table\_name = 'urls' # name of the table to be queried

id\_column = 'id'

column\_2 = 'title'

column\_3 = 'url'

column\_6= 'visit\_count'

conn1 = sqlite3.connect(sqlite\_file1)

conn2= sqlite3.connect(sqlite\_file2)

conn2.executescript('drop table if exists total\_visit\_count')

cursor=conn1.execute('SELECT {cnn},{v} FROM {tn} order by {cnn}'.\

format(cnn=column\_3,v=column\_6,tn=table\_name))

conn2.execute('Create table total\_visit\_count(id integer primary key,url varchar(20),visits integer)')

cnt=0

id=1

for row in cursor :

parsed\_uri=urlparse(row[0])

domain='{uri.scheme}://{uri.netloc}/'.format(uri=parsed\_uri)

read\_url=domain

read\_cnt=row[1]

if (cnt==0):

cnt+=1

temp\_url=domain

temp\_cnt=row[1]

else:

if(read\_url==temp\_url):

temp\_cnt=temp\_cnt+read\_cnt

else:

conn2.execute(' insert into total\_visit\_count(id,url,visits) values (?,?,?)',(id,temp\_url,temp\_cnt))

id+=1

temp\_url=read\_url

temp\_cnt=read\_cnt

conn2.execute(' insert into total\_visit\_count(id,url,visits) values (?,?,?)',(id,temp\_url,temp\_cnt))

fh=open("Report.txt","a")

cursor1=conn2.execute("select \* from total\_visit\_count")

fh.write("Serial number\t\tURL\t\tMax Count\n")

for row in cursor1:

#fh.write("%s\n" % str(row))

fh.write("%s\t\t%s\t\t%s\n" % (str(row[0]),row[1],str(row[2])))

cursor2=conn2.execute('select id, url,max(visits) from total\_visit\_count')

fh.write("--------URL which is the most visited--------\n")

for row in cursor2:

fh.write("%s\t\t%s\t\t%s\n" % (row[0],row[1],row[2]))

#fh.write("%s\n" % str(all\_rows))

fh.close()

conn2.commit()

conn1.close()

conn2.close()

else:

fh=open("Report.txt","a")

6.4.4 whats app function

def whatsapp(self,state):

if self.checkBox2.isChecked():

sqlite\_file1 = 'msgstore.db'

sqlite\_file3 = 'wa.db'

sqlite\_file2 = 'owners.sqlite' # name of the sqlite database file

table\_name2 = 'wa\_contacts' # name of the table to be queried

table\_name1 = 'messages' # name of the table to be queried

#id\_column = 'id'

column\_1 = 'key\_remote\_jid'

column\_2 = 'wa\_name'

#column\_6= ''

conn1 = sqlite3.connect(sqlite\_file1)

conn3= sqlite3.connect(sqlite\_file3)

conn2 = sqlite3.connect(sqlite\_file2)

conn2.executescript('drop table if exists whatsapp')

conn2.executescript('drop table if exists whatsapp\_final')

cursor=conn1.execute("SELECT {cnn} FROM {tn} where {cnn} like '%@s%'order by {cnn}".\

format(cnn=column\_1,tn=table\_name1))

conn2.execute('Create table whatsapp(id integer,count integer)')

conn2.execute('Create table whatsapp\_final(id integer,count integer)')

id1=1

for row in cursor :

string1=row[0]

str1=string1.split("@")

string1=str1[0]

#print string1

conn2.execute(' insert into whatsapp(id,count) values (?,?)',(string1,id1))

cursor=conn2.execute('select \* from whatsapp')

cnt=0

for row in cursor:

read\_num=row[0]

read\_cnt=row[1]

if (cnt==0):

cnt+=1

temp\_num=row[0]

temp\_cnt=row[1]

else:

if(read\_num==temp\_num):

temp\_cnt=temp\_cnt+read\_cnt

else:

conn2.execute(' insert into whatsapp\_final(id,count) values (?,?)',(temp\_num,temp\_cnt))

temp\_num=read\_num

temp\_cnt=read\_cnt

conn2.execute(' insert into whatsapp\_final(id,count) values (?,?)',(temp\_num,temp\_cnt))

#conn2.execute('select \* from whatsapp\_final order by count')

fh=open("Report.txt","a")

cursor1=conn2.execute("select \* from whatsapp\_final")

fh.write("--------Contacted people on Whatsapp--------\n")

fh.write("Id\t\tCount\n")

for row in cursor1:

#fh.write("%s\n" % str(row))

fh.write("%s\t\t%d\n" % (row[0],row[1]))

conn2.commit()

conn1.close()

conn2.close()

6.4.5 Jpeg files function

def jpg\_file(self,state):

if self.checkBox3.isChecked():

conn = sqlite3.connect("owners.sqlite")

curs = conn.cursor()

curs.execute("drop table if exists top\_jpg\_photos")

curs.execute("CREATE TABLE top\_jpg\_photos ( permissions TEXT, Description TEXT, Description2 TEXT,size integer, date TEXT, time integer,name TEXT)")

reader = csv.reader(open('topjpg.csv', 'r'), delimiter=',')

for row in reader:

to\_db = [unicode(row[0], "utf8"), unicode(row[1], "utf8"), unicode(row[2], "utf8"), unicode(row[3], "utf8"), unicode(row[4], "utf8"), unicode(row[5], "utf8"), unicode(row[6], "utf8")]

curs.execute("INSERT INTO top\_jpg\_photos (permissions, Description, Description2 ,size, date, time,name) values (?,?,?,?,?,?,?);", to\_db)

cursor=curs.execute("select \* from top\_jpg\_photos")

fh=open("Report.txt","a")

fh.write("################TOP 100 JPG FILES USED RECENTLY#################\n")

for row in cursor:

fh.write("%s\t%s\t%s\t%s\t%s\t%s\t%s\n" % (row[0],row[1],row[2],row[3],row[4],row[5],row[6]))

fh.close()

conn.commit()

conn.close()

6.4.6 call log function

def call\_logs(self,state):

if self.checkBox4.isChecked():

sqlite\_file1 = 'logs.db'

table\_name = 'logs'

sqlite\_file2 = 'owners.sqlite'

id\_column = '\_id'

column\_1 = 'number'

column\_2 = 'type'

column\_3 = 'logtype'

column\_4 = 'geocoded\_location'

column\_5 = 'frequent'

column\_6 = 'name'

conn1 = sqlite3.connect(sqlite\_file1)

conn2 = sqlite3.connect(sqlite\_file2)

#conn2.execute('drop table if exists call\_logs')

conn2.execute('drop table if exists call\_logs\_final')

conn2.execute('create table call\_logs(id integer primary key,number varchar(15),call\_type integer,geocode varchar(30),count integer,name varchar(20) )')

conn2.execute('create table call\_logs\_final(id integer primary key,number varchar(15),call\_type integer,geocode varchar(30),count integer,name varchar(20) )')

cursor=conn1.execute('select {id\_1},{cn\_1},{cn\_2},{cn\_4},{cn\_5},{cn\_6} from {tn} where {cn\_3}=100; '.\

format(id\_1=id\_column,tn=table\_name, cn\_1=column\_1,cn\_2=column\_2,cn\_3=column\_3,cn\_4=column\_4,cn\_5=column\_5,cn\_6=column\_6))

id1=1

for row in cursor:

num=row[1]

if num.startswith("+"):

num=num[3:]

if num.startswith("0"):

num=num[1:]

conn2.execute(' insert into call\_logs(id,number,call\_type,geocode,count,name) values (?,?,?,?,?,?)',(id1,num,row[2],row[3],row[4],row[5]))

id1+=1

cursor=conn2.execute("select \* from call\_logs order by number")

cnt=0

id2=1

for row in cursor:

num=row[1]

read\_num=num

read\_cnt=row[4]

read\_call\_type=row[2]

read\_geocode=row[3]

read\_name=row[5]

if (cnt==0):

cnt+=1

temp\_num=num

temp\_cnt=row[4]

temp\_call\_type=row[2]

temp\_geocode=row[3]

temp\_name=row[5]

else:

if(read\_num==temp\_num):

temp\_cnt=temp\_cnt+read\_cnt

else:

conn2.execute(' insert into call\_logs\_final(id,number,call\_type,geocode,count,name) values (?,?,?,?,?,?)',(id2,temp\_num,temp\_call\_type,temp\_geocode,temp\_cnt,temp\_name))

id2+=1

temp\_num=read\_num

temp\_cnt=read\_cnt

temp\_call\_type=read\_call\_type

temp\_geocode=read\_geocode

temp\_name=read\_name

id2+=1

conn2.execute(' insert into call\_logs\_final(id,number,call\_type,geocode,count,name) values (?,?,?,?,?,?)',(id2,temp\_num,temp\_call\_type,temp\_geocode,temp\_cnt,temp\_name))

fh=open("Report.txt","a")

fh.write("#################CALL LOGS########################\n")

cursor=conn2.execute("select \* from call\_logs\_final")

for row in cursor:

fh.write("%s\t%s\t%s\t%s\t%s\n" % (row[1],row[2],row[3],row[4],row[5]))

conn2.execute('drop table if exists call\_logs')

conn1.close()

conn2.commit()

conn2.close()

6.4.7 whats app SMS join code

def wa\_sms\_join(self,state):

if self.checkBox5.isChecked():

sqlite\_file3 = 'mmssms.db'

table\_name1='threads'

table\_name2 = 'sms'

id\_column = '\_id'

column\_1 = 'thread\_id'

column\_2 = 'address'

column\_3= 'snippet'

column\_4='message\_count'

#-------------------------

sqlite\_file1 = 'msgstore.db'

sqlite\_file2 = 'owners.sqlite'

table\_name3 = 'frequents'

#id\_column = '\_id'

column\_5 = 'jid'

#column\_3 = 'message\_count'

#---------------------------------

table\_name6 = 'Whatsapp\_msg\_count'

table\_name4 = 'sms\_count'

table\_name5 = 'wa\_sms\_count'

#no1='jid'

#no2='address'

column\_6='count'

#column2='message\_count'

conn1 = sqlite3.connect(sqlite\_file1)

conn2 = sqlite3.connect(sqlite\_file2)

conn3 = sqlite3.connect(sqlite\_file3)

#-------------------------------------------------sms----------------------------------

conn2.execute('drop table if exists sms\_count')

cursor=conn3.execute('select distinct {cnn},{cnn2},{cnn3},{cnn4} from {tn1} inner join {tn2} on threads.\_id=sms.thread\_id order by {cnn4} desc;'.\

format(cnn=column\_1,cnn2=column\_2,cnn3=column\_3,cnn4=column\_4,tn1=table\_name1,tn2=table\_name2))

conn2.execute('Create table sms\_count(id integer primary key autoincrement,address varchar(20),snippet varchar(100), message\_count integer)')

id1=1

for row in cursor :

string1=row[1]

if string1.startswith("+"):

string1=string1[3:]

if string1.startswith("0"):

string1=string1[1:]

if string1.startswith("+"):

string1=re.sub("\\D", "", string1)

conn2.execute(' insert into sms\_count(id,address,snippet,message\_count) values (?,?,?,?)',(id1,string1,row[2],row[3]))

id1+=1

cursor2=conn2.execute('select max(message\_count) as id,address,snippet,message\_count from sms\_count')

#----------------------------------------------------WA join-----------------------------------

conn2.execute('drop table if exists Whatsapp\_msg\_count')

conn2.execute('Create table Whatsapp\_msg\_count(id integer primary key,jid integer,count integer)')

cursor=conn1.execute('SELECT {id\_1},{cn},{cn\_2} FROM {tn} order by {cn\_2} desc'.\

format(id\_1=id\_column,tn=table\_name3, cn=column\_5,cn\_2=column\_4))

id2=1

for row in cursor:

string1=row[1]

spl=string1.split("@")

num=spl[0]

num=num[2:]

conn2.execute(' insert into Whatsapp\_msg\_count(id,jid,count) values (?,?,?)',(id2,num,row[2]))

id2+=1

#--------------------------WA-SMS-Join---------------------------------------------

conn2.execute('drop table if exists wa\_sms\_count')

conn2.execute('drop table if exists wa\_sms\_count\_final')

cursor=conn2.execute('select {n1},coalesce({cnn1},{cnn2}) as merged\_count from {tn1} left outer join {tn2} on {tn1}.{n1}={tn2}.{n2} union select {n2},coalesce({cnn2},{cnn1}) as merged\_count from {tn2} left outer join {tn1} on {tn2}.{n2}={tn1}.{n1};'.\

format(n1=column\_5,n2=column\_2,cnn1=column\_6,cnn2=column\_4,tn1=table\_name6,tn2=table\_name4))

conn2.execute('Create table wa\_sms\_count(id integer primary key,address varchar(20),message\_count integer)')

conn2.execute('Create table wa\_sms\_count\_final(id integer primary key,address varchar(20),message\_count integer)')

id1=1

for row in cursor:

conn2.execute(' insert into wa\_sms\_count(id,address,message\_count) values (?,?,?)',(id1,row[0],row[1]))

id1=id1+1

cursor1=conn2.execute('select \* from {tn3} order by {n2}'.\

format(n2=column\_2,tn3=table\_name5))

cnt=0

id2=1

for row in cursor1 :

domain=row[1]

read\_no=domain

read\_cnt=row[2]

if (cnt==0):

cnt+=1

temp\_no=domain

temp\_cnt=row[2]

else:

if(read\_no==temp\_no):

temp\_cnt=temp\_cnt+read\_cnt

else:

conn2.execute(' insert into wa\_sms\_count\_final(id,address,message\_count) values (?,?,?)',(id2,temp\_no,temp\_cnt))

id2+=1

temp\_no=read\_no

temp\_cnt=read\_cnt

fh=open("Report.txt","a")

fh.write("#################WHATSAPP AND SMS########################\n")

cursor=conn2.execute("select \* from wa\_sms\_count\_final order by message\_count desc")

for row in cursor:

fh.write("%s\t%d\n" % (row[1],row[2]))

conn2.commit()

conn2.close()

conn1.close()

conn3.close()

6.4.8 SMS count function

def sms\_count(self,state):

if self.checkBox6.isChecked():

sqlite\_file1 = 'mmssms.db'

sqlite\_file2 = 'owners.sqlite'

table\_name1='threads'

table\_name2 = 'sms'

id\_column = '\_id'

column\_1 = 'thread\_id'

column\_2 = 'address'

column\_3= 'snippet'

column\_4='message\_count'

conn1 = sqlite3.connect(sqlite\_file1)

conn2= sqlite3.connect(sqlite\_file2)

conn2.execute('drop table if exists sms\_count')

cursor=conn1.execute('select distinct {cnn},{cnn2},{cnn3},{cnn4} from {tn1} inner join {tn2} on threads.\_id=sms.thread\_id order by {cnn4} desc;'.\

format(cnn=column\_1,cnn2=column\_2,cnn3=column\_3,cnn4=column\_4,tn1=table\_name1,tn2=table\_name2))

conn2.execute('Create table sms\_count(id integer primary key autoincrement,address varchar(20),snippet varchar(100), message\_count integer)')

id1=1

for row in cursor :

string1=row[1]

if string1.startswith("+"):

string1=string1[3:]

if string1.startswith("0"):

string1=string1[1:]

conn2.execute(' insert into sms\_count(id,address,snippet,message\_count) values (?,?,?,?)',(id1,string1,row[2],row[3]))

id1+=1

fh=open("Report.txt","a")

fh.write("#################SMS########################\n")

cursor=conn2.execute("select \* from sms\_count order by message\_count desc limit 5")

for row in cursor:

fh.write("%s\t%s\t%d\n" % (row[1],row[2],row[3]))

conn2.commit()

conn2.close()

conn1.close()

6.4.9 Facebook gmail

def fb\_gmail(self,state):

if self.checkBox7.isChecked():

sqlite\_file1="mailstore.db"

sqlite\_file2="owners.sqlite"

table1\_name="messages"

column\_1="fromAddress"

column\_2="toAddresses"

column\_3="subject"

column\_4="snippet"

column\_5="joinedAttachmentInfos"

conn1 = sqlite3.connect(sqlite\_file1)

conn2= sqlite3.connect(sqlite\_file2)

conn2.execute("drop table if exists fb\_gmail")

conn2.execute("create table fb\_gmail(fromAddress varchar(50),toAddresses varchar(50), subject varchar(50),snippet varchar(20))")

cursor=conn1.execute("select {cnn1},{cnn2},{cnn3},{cnn4} from {tn1} where {cnn5} like '%.facebook\_%';".\

format(cnn1=column\_1,cnn2=column\_2,cnn3=column\_3,cnn4=column\_4,cnn5=column\_5,tn1=table1\_name))

for row in cursor:

conn2.execute(' insert into fb\_gmail(fromAddress,toAddresses,subject,snippet) values (?,?,?,?)',(row[0],row[1],row[2],row[3]))

fh=open("Report.txt","a")

fh.write("#################facebook to gmail########################\n")

cursor=conn2.execute("select \* from fb\_gmail")

for row in cursor:

fh.write("%s\t%s\t%s\t%s\n" % (row[0],row[1],row[2],row[3]))

conn2.commit()

conn2.close()

conn1.close()

6.4.10 Whats app to gmail

def wa\_gmail(self,state):

if self.checkBox8.isChecked():

sqlite\_file1="mailstore.db"

sqlite\_file2="owners.sqlite"

table1\_name="messages"

column\_1="fromAddress"

column\_2="toAddresses"

column\_3="subject"

column\_4="snippet"

column\_5="joinedAttachmentInfos"

conn1 = sqlite3.connect(sqlite\_file1)

conn2= sqlite3.connect(sqlite\_file2)

conn2.execute("drop table if exists wa\_gmail")

conn2.execute("create table wa\_gmail(fromAddress varchar(50),toAddresses varchar(50), subject varchar(50),snippet varchar(20))")

cursor=conn1.execute("select {cnn1},{cnn2},{cnn3},{cnn4} from {tn1} where {cnn5} like '%-WA%';".\

format(cnn1=column\_1,cnn2=column\_2,cnn3=column\_3,cnn4=column\_4,cnn5=column\_5,tn1=table1\_name))

for row in cursor:

conn2.execute(' insert into wa\_gmail(fromAddress,toAddresses,subject,snippet) values (?,?,?,?)',(row[0],row[1],row[2],row[3]))

fh=open("Report.txt","a")

fh.write("#################whatsapp to gmail########################\n")

cursor=conn2.execute("select \* from wa\_gmail")

for row in cursor:

fh.write("%s\t%s\t%s\t%s\n" % (row[0],row[1],row[2],row[3]))

conn2.commit()

conn2.close()

conn1.close()

6.4.11 Messenger function

def messenger\_contacts(self,state):

if self.checkBox9.isChecked():

sqlite\_file1 = 'threads\_db2'

sqlite\_file2 = 'owners.sqlite'

table\_name1 = 'threads'

table\_name2 = 'messages'

column\_1 = 'sender'

column\_2 = 'thread\_key'

column\_3 = 'snippet'

column\_4 = 'approx\_total\_message\_count'

conn1 = sqlite3.connect(sqlite\_file1)

conn2 = sqlite3.connect(sqlite\_file2)

conn2.execute('drop table if exists messenger\_total\_users')

conn2.execute('drop table if exists messenger\_user\_names')

fh=open("Report.txt","a")

fh.write("#################CONTACTED PEOPLE ON MESSENGER APP########################\n")

#id2=1

thread='\0'

cursor1=conn1.execute("select distinct {cn\_1},{tn1}.{cn\_2},{cn\_3}, {cn\_4} from {tn1} inner join {tn2} on {tn1}.{cn\_2}={tn2}.{cn\_2} where {cn\_1} not null AND {tn1}.{cn\_2} not like 'GROUP%' order by {cn\_4} desc;".\

format(tn1=table\_name1,tn2=table\_name2, cn\_2=column\_2, cn\_1=column\_1,cn\_3=column\_3, cn\_4=column\_4))

for row in cursor1:

string1=row[1]

spl=string1.split(":")

thread=spl[2]

break

conn2.execute('create table messenger\_total\_users(sender varchar(30),thread\_key varchar(100), snippet varchar(100), message\_count integer)')

conn2.execute('create table messenger\_user\_names(user\_key varchar(100),name varchar(30))')

cursor2=conn1.execute("select user\_key,name from thread\_users")

for row in cursor2:

string2=row[0]

spl2=string2.split(":")

thread2=spl2[1]

name=row[1]

if thread==thread2:

name\_acc = row[1]

fh.write("ACCOUNT BELONG TO %s" % (name\_acc))

conn2.execute('insert into messenger\_user\_names(user\_key,name) values(?,?)',(thread2,row[1]))

for row in cursor1:

string3=row[0]

spl3=string3.split('"')

thread3=spl3[5].split(":")

thread3=thread3[1]

if thread3!=thread:

conn2.execute('insert into messenger\_total\_users(sender,thread\_key,snippet,message\_count) values(?,?,?,?)',(thread3,row[1],row[2],row[3]))

cursor3=conn2.execute("select sender,name,message\_count from messenger\_total\_users inner join messenger\_user\_names on messenger\_total\_users.sender = messenger\_user\_names.user\_key")

for row in cursor3:

fh.write("\n%s\t%s\t%d\n" % (row[0],row[1],row[2]))

conn2.commit()

conn1.close()

conn2.close()

6.4.12 Report opening function

def open\_file(self):

os.system("chmod 777 %s/kkgm/%s/Report.txt" % (path,date\_time))

#os.system("enscript /home/mudita/kkgm/Report.txt -o - |ps2pdf -Report.pdf")

status,output = commands.getstatusoutput("enscript -p output.ps %s/kkgm/%s/Report.txt |ps2pdf output.ps Report.pdf" % (path,date\_time))

if sys.platform=="linux2":

subprocess.call(["xdg-open","Report.pdf"])

else:

os.startfile("Report.pdf")

**Chapter 7: SOFTWARE TESTING**

**7.1 System Test Objective**

The objective of System testing is to describe the test plan, task schedule of our system to be tested as a whole.

These objectives are as follows:

1. It must ensure that all the modules are working correctly as per user’s requirements.

2. Performance requirements and functional requirements are met.

3. The system should work properly in the environment where it would be last deployed.

4. It should be fault tolerant and aim at data consistency.

5. Maintenance of software must be easy and cheap.

6. The software must be reliable and robust.

Full set of system tests will be performed including:

1. **Functional testing**- Test the functioning of every module.

2. **Performance Testing**- Test the overall performance of the system in terms of response time, efficiency.

3. **Regression testing**- This testing will find the bugs which may get introduced accidentally because of the new changes or modification.

4. **Unit testing**- This type of testing will verify the correctness of every module.

5. **Integration testing**- This will test the individual modules when combined in a group .

**7.2 Approach**

In our approach it is ensured that all the databases are extracted correctly. Major testing is done on the extraction part of databases so that required information after querying gets generated in the final report. Unit testing is performed on every module. Also integration testing is performed after integrating these unit modules. Focus on regression testing is also made as data extraction may every time have new appended data to the previous data source.

In short every part of the system is tested to check if it is working correctly.

**7.3 Test tools**

Our user guided software is tested manually.

**7.4 Test Deliverables**

1. System Test plan

2. System test case specification

**7.5 Items to be tested**

* Mobile phone
* Laptop
* Our software to be developer

**7.6 Test cases**

|  |  |
| --- | --- |
| Test case 1 | Mobile phone connection with laptop via  USB cable |
| Scenario | User connects the mobile phone to the laptop |
| Input | Mobile phone connection |
| Output | Phone connected successfully |
| Expected output | Phone connected successfully |

|  |  |
| --- | --- |
| Test case 2 | Detect the target mobile device |
| Scenario | On the connection of mobile device, press  OK to verify if the connected device is the target device |
| Input | Press OK button |
| Output | Model number and build number displayed |
| Expected output | Model number and build number displayed |

|  |  |
| --- | --- |
| Test case 3 | Detect the target mobile device |
| Scenario | On the connection of mobile device, press  cancel button if the connected device is not the target device |
| Input | Press cancel button |
| Output | Goes to welcome page |
| Expected output | Goes to welcome page |

|  |  |
| --- | --- |
| Test case 4 | Data extraction |
| Scenario | Press OK button for extraction of databases |
| Input | Press OK button |
| Output | “Extracted successfully” displayed |
| Expected output | “Extracted successfully” displayed |

|  |  |
| --- | --- |
| Test case 5 | Selected option’s output display |
| Scenario | List of radio buttons are provided. Output of  only selected radio buttons is to be generated in report. |
| Input | Select the desired radio buttons |
| Output | Output of only selected radio buttons is  generated in report successfully |
| Expected output | Output of only selected radio buttons is  generated in report successfully |

|  |  |
| --- | --- |
| Test case 6 | Selected option’s output display |
| Scenario | Nothing is generated in report if radio buttons  are kept unselected |
| Input | No radio button is selected |
| Output | Nothing is generated in the report |
| Expected output | Nothing is generated in the report |

|  |  |
| --- | --- |
| Test case 7 | Phone disconnected in between extraction |
| Scenario | During extraction the phone is disconnected |
| Input | Detach the phone from laptop |
| Output | Incomplete extraction of databases |
| Expected Output | Incomplete extraction of databases |

|  |  |
| --- | --- |
| Test case 8 | Phone disconnected after successful  extraction |
| Scenario | After successful attraction of databases, the  mobile device is detached from the laptop. |
| Input | Mobile device is detached from laptop |
| Output | Proceed with further desired operations |
| Expected Output | Proceed with further desired operations |

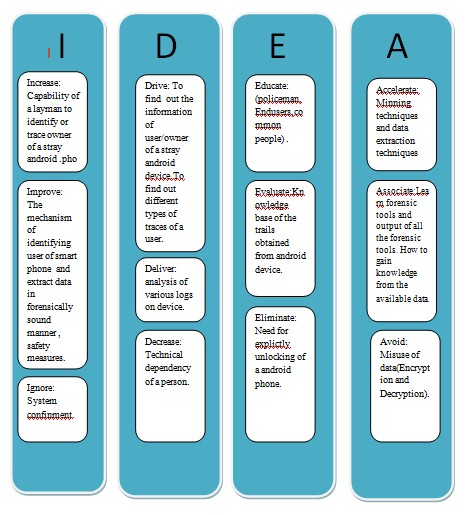
**Chapter 8: CONCLUSION**

We presented existing methods to analyze mobile phones and pointed out their advantages and disadvantages. Since these methods either violate forensic principles or necessitate advanced knowledge to perform the analysis, we have presented a guided software which enables automated analysis. This software accesses the device via the Android Debug Bridge in order to retrieve a copy of selected SQLite databases. Subsequently, the SQLite databases are parsed and data is extracted and finally transformed into a report. During the development

of this software main forensic principles have been taken into consideration.

**Appendix A: Laboratory assignments on project**

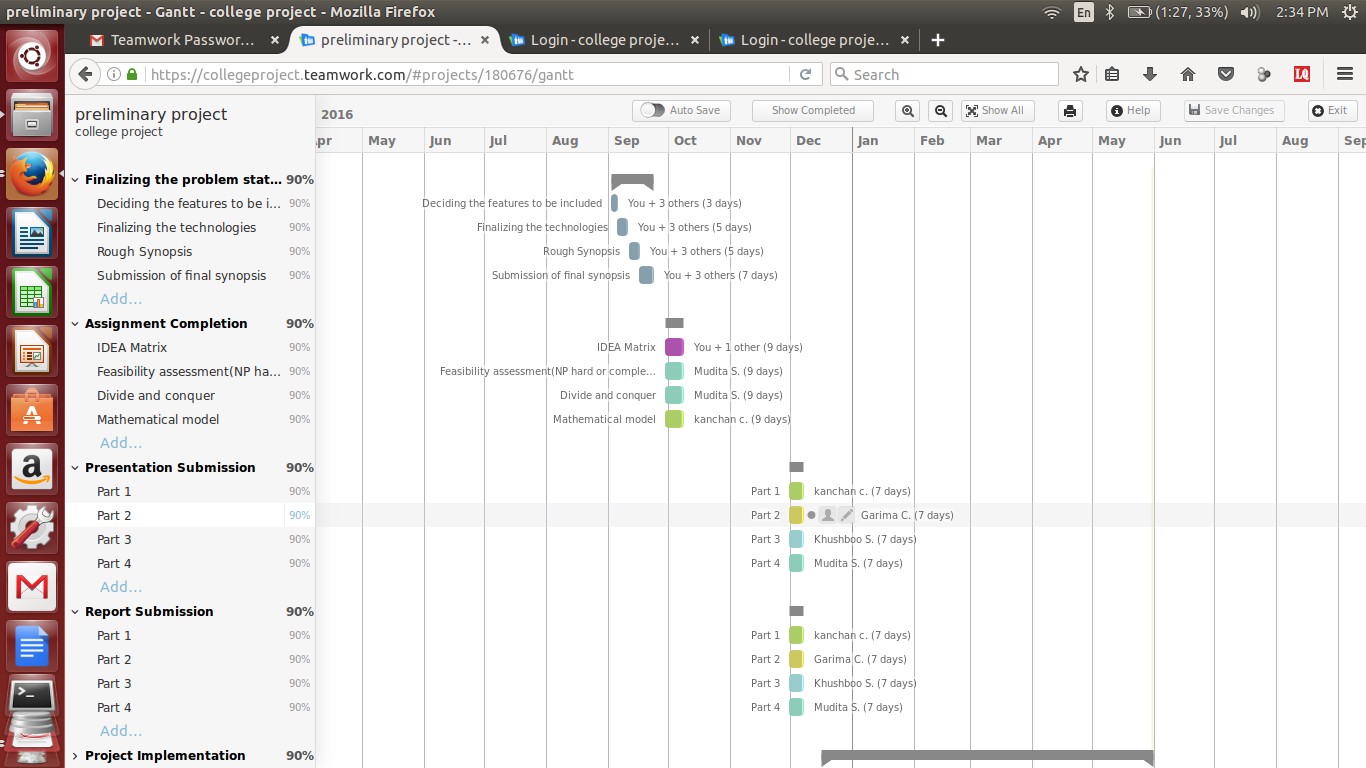
**Assignment 1:Idea Matrix**

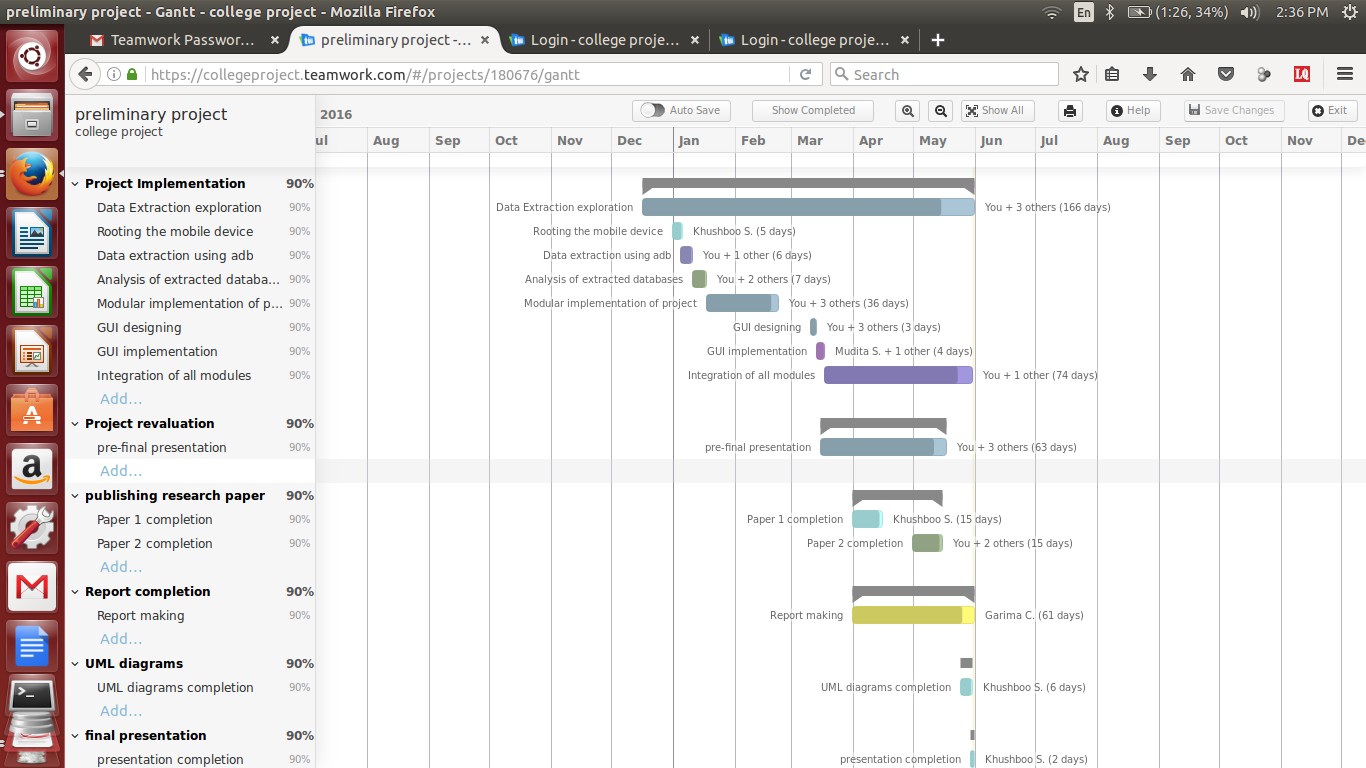


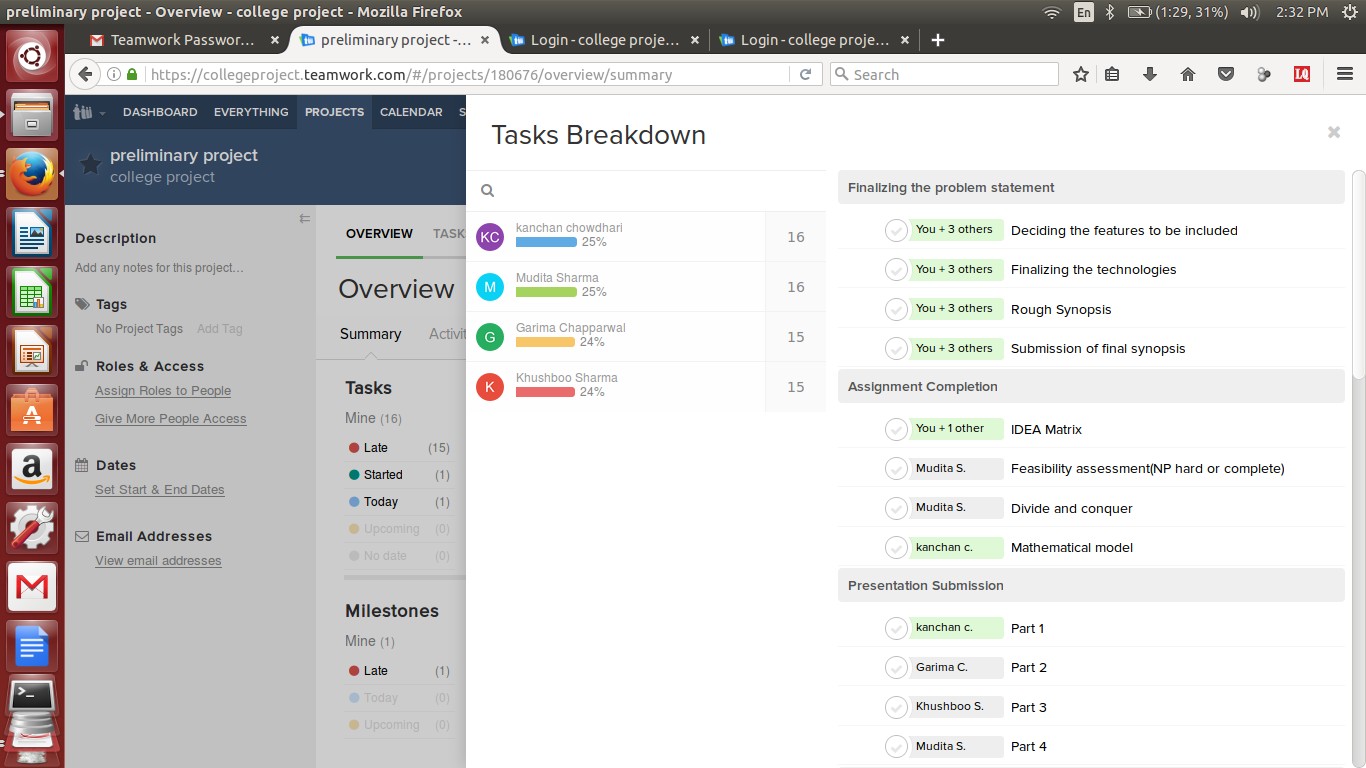
**Assignment 2 Project problem statement: Feasibility assessment using NP Hard, NP Complete or satisfiability issues, using modern algebra and/or relevant mathematical model**

On an Android based systems there are various logs and files present that can be useful in identifying the user trails. For a perspective that at a given scope, consideration of all possible such sources make the project statement in an intractable state. This in turn results in the NP hard state. However by scoping the log and file sources as per the usability and time constraints, we have turned it into a tractable state i.e NP Complete.

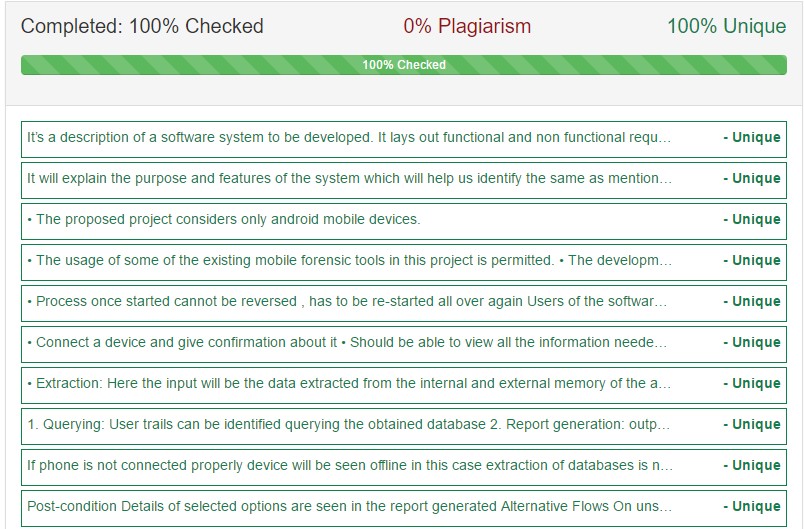
**Appendix C:Project Planner**

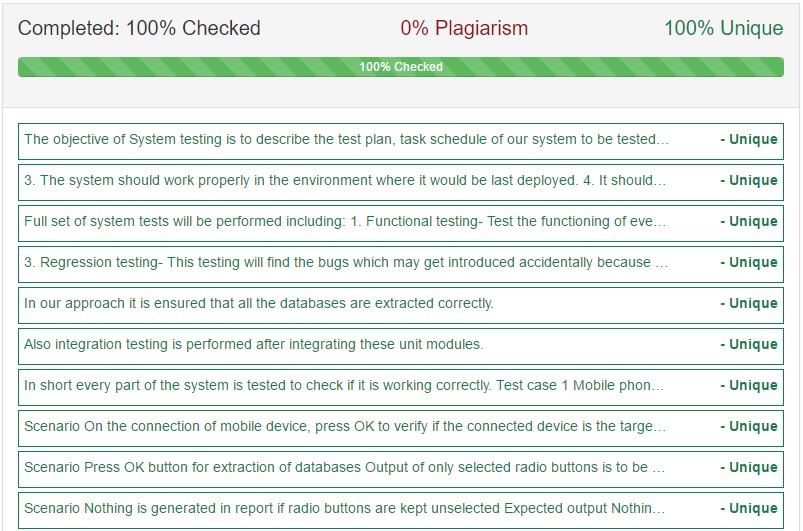


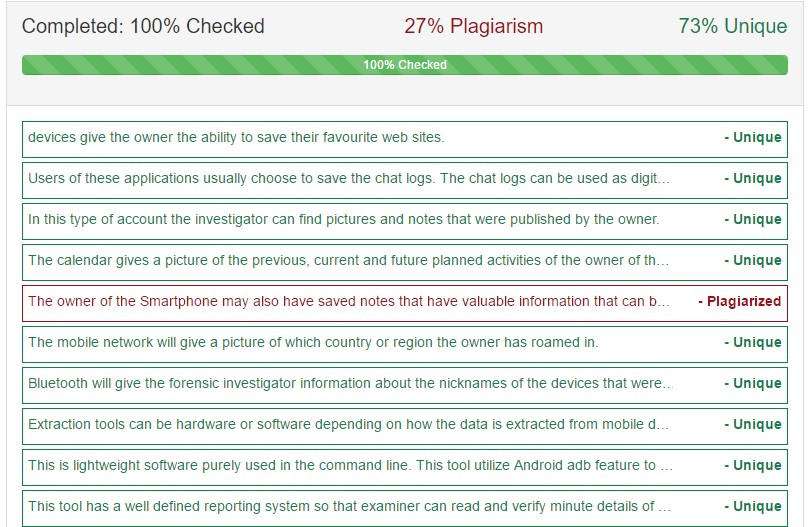




**Appendix D:Plagarism**







**Appendix E: Glossary**

**Android forensics**: Extracting data from smart phone and recovering the evidences in forensically sound manner.

**Evidence**: Available body of facts or information indicating whether a belief is a true or valid.

**Trails**: disparate findings of forensic investigations which are combined to get conclusive investigation report.

**Smartphone**: It is a mobile phone built on a mobile operating system, with more advanced computing capability and connectivity than a feature phone.

**REFRENCES:**

1.Jeff Lessard, Gary C. Keseler, **“**Android forensics simplifying cell phone examination**”,** Small Scale Digital Device Forensics Journal, Vol. 4 No. 1

2. Felix Freiling, Michael Spreitzenbarth, Sven Schmitt, **“**Forensic analysis of smart phones: The android data extractor lite (ADEL)”, ADFSL Conference on Digital Forensics, Security and Law, 2011

3. Khawla Abdulla Alghafli1, Andrew Jones1, Thomas Anthony Martin, “ Guidelines

for the digital forensic processing of smart phones” ,Australian conference 2011

4. Timothy Vidas et al. (Vidas, Zhang, &Christin, 2011) , “Forensic analysis of the android file system YAFFS2” , Australian conference 2011

**WEB REFRENCES:**

1. [https://www.andriller.com/decoders -](https://www.andriller.com/decoders)A guide to find out database path in an android file system

2. [https://www.xdadevelopers.com-](https://www.andriller.com/decoders) A guide to Phone rooting and ADB rooting

3. [http://www.tutorialspoint..com/pyqt - A](https://www.andriller.com/decoders) guide for developing GUI

4. <http://freeandroidforensics.blogspot.in/2014/0> 8/imaging-android-device.html -Live imaging of an android device

5. <http://freeandroidforensics.blogspot.in/2014/0> 8/examining-image.html - Live imaging of an android device

6. https:/[/www.youtube.com/watch](http://www.youtube.com/watch)?v=bfs25Nijw -Rooting an android device using

cloud based method