**INTRODUCTION**

# 1.1 PROBLEM DEFINITION

We can’t access the information wherever we want with our own requirements**.** The notion of this project is to access information when the mobile is not beside. It comforts with the required information from any location.

# 1.2 NEED FOR THE PROJECT

Once a user forgets his mobile and need to know his mobile current status or need to surveillance his mobile in the sense it is possible to do such things without any human action. It is not possible to get the mobile information without help of some humans and it is not possible to change the profile of the mobile without user interaction.

We can’t make remote access to get prior information through the SMS for our own mobile device. At present we access all the information only with the help of mobile or we should keep the contact details with our dictionary or dairy. We can’t access the information wherever we want for our own requirement.

**1.3 OVERVIEW OF THE PROJECT**

This mobile surveillance application is to create a new way of approach in android mobile platform that discovers various excited applications of SMS over its traditional text messaging application. The Short Message Service (SMS) allows text-based messages to be send from mobile telephones on a GSM network. Many applications need the ability to do real-time notification when events occur. Often the people who need to be kept aware of events are in a remote location. Here the same is giving the notification to the user whenever they want to communicate with their mobile virtually. It shows how various fea*t*ures in android mobiles can be automated by SMS. By this application user can perform various operations in its mobile even if mobile is very far from him , like by sending a single text message we can fetch and retrieve our contact numbers, auto respond to the incoming messages, send SMS from our remote mobile, fetching SIM and mobile details used for GSM network. Convenience to the user, security and efficiency are main issues that are considered.

**1.4 OBJECTIVE OF THE PROJECT**

Here objective defines whenever you want to access your device virtually and get some details dynamically. This app will satisfy your need for your convenience. To find the contact log from the Android mobile devices whenever we required the contact and change the profile mode(such as normal, airplane and silent) of the device for our convenient. This application proved to get the recent call logs details when you sent a message to the device. Then you may want to see contact list those which are think from your mind alphabetical contacts you can get from your mobile.

**1.5 SCOPE OF THE PROJECT**

Main scope of the projects is to give the surveillance to our mobile device through the SMS. Give the SMS details to the destination mobile and it will compare the SMS information and it will send the more relevant information to the user. We can access our contact details whenever we want through remote access. And also to provide the security for our android devices because we should change the profile mode through remote access.

**LITERATURE SURVEY**

Literature survey is the documentation of a comprehensive review of the published and unpublished work from secondary sources data in the areas of specific interest to the researcher. The library is a rich storage base for secondary data and researchers used to spend several weeks and sometimes months going through books, journals, newspapers, magazines, conference proceedings, doctoral dissertations, master's theses, government publications and financial reports to find information on their research topic. With computerized databases now readily available and accessible the literature search is much speedier and easier and can be done without entering the portals of a library building.

**2.1 CONTEXT-BASED ACCESS CONTROL FOR RIDESHARING**

**SERVICE**

The paper describes a context-based access control model for ridesharing service. Ridesharing is a shared use of a car by the driver and one or more passengers for a joint trip. The service is based on the smart space concept. For this purpose the Smart-M3 platform is used. Currently the Smart-M3 platform doesn’t have an appropriate access control mechanism meeting the following requirements: supporting a flexible, descriptive and well-defined policy language and taking into consideration the context information. Therefore, the usage of the context-based access control model has been proposed. This model is built as a combination of the role-based and attribute-based access control models. It uses roles, which are assigned dynamically based on the user’s context, and meets the requirements to the access control in the smart space. An analysis of information transfer through the ridesharing service modules is used for defining the user’s context. The model has been implemented within an access control broker. This controls the access to the smart space resources.

**DRAWBACK**

For the wide range of service, it was needed to provide a mechanism that allows ridesharing service users to restrict access for their private information.

**2.2 A CONTEXT-RISK-AWARE ACCESS CONTROL MODEL FOR**

**UBIQUITIOUS ENVIRONMENTS**

This paper reports our ongoing work to design a Context-Risk-Aware Access Control (CRAAC) model for Ubiquitous Computing (UbiComp) environments. CRAAC is designed to augment flexibility and generality level of assurance play a key role in CRAAC. Through risk assessment, resources are classified into groups according to their sensitivity levels and potential impacts should any unauthorized access occurs. The identified risks are mapped onto their required assurance levels, called Object Level of Assurance (OLoA). Upon receiving an object access request, the requester’s run-time contextual information is assessed to established a Requester’s Level of Assurance (RLoA) denoting the level of confidence in identifying that requester. The access request is granted if RLoA≥OLoA. This paper describes the motivation for, and the design of, the CRAAC model, and reports a case study to further illustrate the model.

**DRAWBACK**

High level algorithms needed and each algorithm has their individual drawbacks.

**2.3 DRBAC: DISTRIBUTED ROLE-BASED ACCESS CONTOL FOR**

**DYNAMIC COALITION ENVIRONMENTS**

Distributed Role-Based Access Control (dRBAC) is a scalable, decentralized trust-management and access-control mechanism for systems that span multiple administrative domains. dRBAC utilizes PKI identities to define trust domains, roles to define controlled activities, and role delegation across domains to represent permissions to these activities. The mapping of controlled actions to roles enables their namespaces to serve as policy roots.

dRBAC distinguishes itself from previous approaches by providing three features: (1) third-party delegation of roles from outside a domain’s namespace, relying upon an explicit delegation of assignment; (2) modulation of transferred permissions using scalar valued attributes associated with roles; and (3) continuous monitoring of trust relationships over long-lived interactions. This paper describes the dRBAC model and its scalable implementation using a graph approach to credential discovery and validation.

**DRAWBACK**

Delegation will permit construction of directory based cache.

**2.4 MODELLLING CONTEXTS IN THE Or-BAC MODEL**

As computer infrastructures become more complex, security models must provide means to handle more flexible and dynamic requirements. In the Organization Based Access Control (Or-BAC) model, it is possible to express such requirements using the notion of context. In Or-BAC, each privilege (permission or obligation or prohibition) only applies in a given context. A context is viewed as an extra condition that must be satisfied to activate a given privilege. In this paper, we present taxonomy of different types of context and investigate the data the information system must manage in order to deal with these different contexts. We then explain how to model them in the Or-BAC model.

**DRAWBACK**

Fail to express large number of different contexts within a single famwork.

**2.5 AN ACCESS CONTROL ARCHITECTURE FOR DISTRIBUTING**

**TRUST IN PERVASIVE COMPUTING ENVIRONMENTS**

Pervasive computing infrastructure is highly distributed and it is essential to develop security mechanism that enhances the security of the system by distributing trust among the various infrastructure components. We present a novel access control architecture explicitly designed to distributed trust that combines threshold cryptography, multi-layer encryption and mediated access to contextual data to support dynamically changing access control permissions. We present several models of our access control infrastructure and evaluated how well each design distributes trust and limits the behavior of our threshold-based access control scheme and evaluated the overhead of each infrastructure model.

**DRAWBACK**

Have to combine multiple sensors reading a difficult process for real time approach.

**2.6 PRACTICAL AND LIGHTWEIGHT DOMAIN ISOLATION ON**

**ANDROID**

In this paper, we introduce a security framework for practical and lightweight domain isolation on Android to mitigate unauthorized data access and communication among applications of different trust levels (e.g., private and corporate). We present the design and implementation of our framework, TrustDroid, which in contrast to existing solution enables isolation at different layers of the Android software stack: (1) at the middleware layer to prevent inter-domain application communication and data access, (2) at the kernel layer to enforce mandatory access control on the file system and on Inter-Process Communication (IPC) channels, and (3) at the network layer to mediate network traffic. For instance, (3) allows network data to be only read by a particular domain, or enables basic context-based policies such as preventing Internet access by untrusted applications while an employee is connected to the company’s network.

Our approach accurately addresses the demands of the business world, namely to isolate data and applications of different trust levels in a practical and lightweight way. Moreover, our solution is the first leveraging mandatory access control with TOMOYO Linux on a real Android device (Nexus One). Our evaluation demonstrates that TrustDroid only adds a negligible overhead, and in contract to contemporary full virtualization, only minimally affects the battery’s life-time.

**DRAWBACK**

Cannot prevent an adversary from compromising the trusted computing base.

**2.7 A SURVEY OF MOVILE MALWARE IN THE WILD**

Mobile malware is rapidly becoming a serious threat. In this paper, we survey the current state of mobile malware in the wild. We analyze the incentives behind 46 pieces of iOS, Android, and Symbian malware that spread in the wild from 2009 to 2011. We also use this data set to evaluate the effectiveness of techniques for preventing and identifying mobile malware. After observing that 4 pieces of malware use root exploits to mount sophisticated attacks on Android phones, we also examine the incentives that cause non-malicious smartphone tinkerers to publish root exploits and survey the availability of root exploits.

**DRAWBACK**

Malwares can use root exploits to circumvent smart phones security mechanism.

**2.8 FLEXIBLE AND FINE-GRAINED MANDATORY ACCESS CONTROL**

**ON ANDROID FOR DIVERSE SECURITY AND PRIVACY POLICIES**

In this paper we tackle the challenge of providing a generic security architecture for the Android OS that can serve as a flexible and effective ecosystem to instantiate different security solutions. In contrast to prior work our security architecture, termed FlaskDroid, provides mandatory access control simultaneously on both Android’s middleware and kernel layers. The alignment of policy enforcement on these two layers is non-trivial due to their completely different semantics. We present an effective policy language (inspired by SELinux) tailored to the specifics of Android’s middleware semantics. We show the flexibility of our architecture by policy-driven instantiation of selected security models such as the existing work Saint as well as a new privacy-protecting, user-defined and fine-grained per-app access control model. Other possible instantiations include phone booth made, or dual persona phone. Finally we evaluate our implementation on SE Android 4.0.4 illustrating its sufficiency and effectiveness.

**DRAWBACK**

When any application is installed, it has been granted all requested permission so error handling code is denied.

**2.9 A STUDY OF ANDROID APPLICATION SECURITY**

The fluidity of application markets complicate smart phone security. Although recent efforts have shed light on particular security issues, there remains little insight into broader security characteristics of smartphone applications. We design and execute a horizontal study of smartphone applications based on static analysis of 21 million lines of recovered code. Our analysis uncovered pervasive use/misuse of personal/phone identifiers, and deep penetration of advertising and analytics networks. However, we did not find evidence of malware or exploitable vulnerabilities in studied applications. We conclude by considering the implications of these preliminary findings and offer directions for future analysis.

**DRAWBACK**

This uncovered pervasive use or misuse of personal phone identifiers and deep penetration of advertising analytics networks.

**2.10 CURBING ANDROID PERMISSION CREEP**

The Android platform has about 130 application level permissions that govern access to resources. The determination of which permissions to request is left solely to the application permission at install time, and permissions are silently enforced at execution time. Although many application make use of a wide range of permissions, we observed that some applications request permission that are not required for the application to execute, and that existing developer APIs make it difficult for developers to align their permission requests with application functionality. In this paper we describe a tool we developed to assist developers in utilizing least privilege.

**DRAWBACK**

There is no requirement for developers to release the source code of android application.

**TABLE 1: Comparison Table For Literature Survey**

|  |  |  |  |
| --- | --- | --- | --- |
| **TITLE** | **METHODLOGY** | **ADVANTAGE** | **PROBLEM** |
| Context-Based Access Control for Ridesharing Service (2013) | Formalize context | Allows users to share their private information only with persons they trust via using the easy configurable preferences | For the wide range of service, it was needed to provide a mechanism that allows ridesharing service users to restrict access for their private information |
| An access control architecture for distributing trust in pervasive computing environment2 | Symmetric key algorithm | Combines the use of threshold cryptography and heterogeneous high level contexts to make an access control decisions | Have to combine multiple sensors reading a difficult process for real time approach |
| **TITLE** | **METHODLOGY** | **ADVANTAGE** | **PROBLEM** |
| drBAC: Distributed Role based Access control for dynamic coalition environments | Proof monitoring and a centralized trust relationship queries | Allows decentralized trust management and access control mechanism for systems that spam multiple administrative domains | Delegation will permit construction of directory based cache |
| Modelling contexts in the Or-BAC model | Temporal and spatial context | Context is viewed as an extra condition and investigate the data that the system manage | Fail to express large number of different contexts within a single framework |
| **TITLE** | **METHODLOGY** | **ADVANTAGE** | **PROBLEM** |
| Practical and Light weight domain isolation on android | 50 apps available from android market and grouped in two domains | Deployed in business transactions and allows employees to get necessary information from company’s network | Cannot prevent an adversary from compromising the trusted computing base |
| A study of android application security | Control flow analysis | Increased security for end users and get rid of existence of dangerous functionality | This uncovered pervasive use or misuse of personal phone identifiers and deep penetration of advertising analystics network |
| **TITLE** | **METHODLOGY** | **ADVANTAGE** | **PROBLEM** |
| Curbing android permission creep | Eclipse api database | Users efficiently make security decisions especially when a warning is difficult to understand | There is no requirement for developers to release the source code of android application |

**2.11 FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

**2.11.1 ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### 2.11.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**2.11.3 SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**SYSTEM DESIGN**

**3.1 EXISTING SYSTEM**

It is not possible to get the mobile information without help of some humans and it is not possible to change the profile of the mobile without user interaction. Once a user forgets his mobile and need to know his mobile current status or need to surveillance his mobile in the sense it is possible to do such things without any human action.

**DISADVANTAGES**

1. We cannot communicate with other mobile without need any intruder.
2. We can’t make remote access the prior information through the SMS for our own mobile device.
3. We change our mobile profile without help of any human action.

**3.2 PROPOSED SYSTEM**

It is a mobile phone based communication application. The Short Message Service (SMS) allows text-based messages to be send from mobile telephones on a GSM network. .Many applications need the ability to do real-time notification when events occur. Often the people who need to be kept aware of events are in a remote location. Here the same is to giving the notification to the user whenever they want to communicate with their mobile virtually. To find the contact lag from the Android mobile devices whenever we required the contact and change the profile mode of the device for our convenient. This application proved to get the recent call logs details when you sent a message to the device. Then you may want to see contact list those which are think from your mind alphabetical contacts you can get from your mobile. And you change the profile mode for your mobile and if you want Switch off your mobile you can. It is used on Business point of view surveillance and Security. Real time example if the user forgot his mobile at his home and context is very important to access the last moment data received by his mobile. Here context is nothing but the surrounding situation. If you need business point of view you need to access the last moment message otherwise to access the recent call logs which may be important or you need to access the contacts in alphabetical order. In that situation you need to give a privacy to your mobile device you can change your mobile to change the profile mode or to lock the screen. If suppose in your mobile is in silent mode you need anybody to want to access your mobile in your mobile surroundings you can change the mobile device into the normal mode.

This application proved to surveillance the user mobile virtually from some other place through SMS commands to get the recent contact details. And we can change the profile mode of the device for need to avoid your device from other human interaction. Then the user needs to get the last messages he can retrieve it through our command. We can access our contact details whenever we want through remote access. This application provide the security for our android devices because we should change the profile mode through remote access. The main objective of this application is to give the surveillance to our android mobile.

User must register the security key and relevant details before accessing the device using commands. User sends a command to the mobile device which is already coded using Eclipse IDE. By sending respective syntax we can retrieve recent call logs , messages and contacts in the mobile device. Also we can change the profile mode of a device. Audio Manager is used to change the mode of the device. When the user sends a command to change the profile mode, first the syntax is verified and the command is passed to the audio manager. When the syntax is satisfied the audio manager class is enabled and the mode of the device is changed.

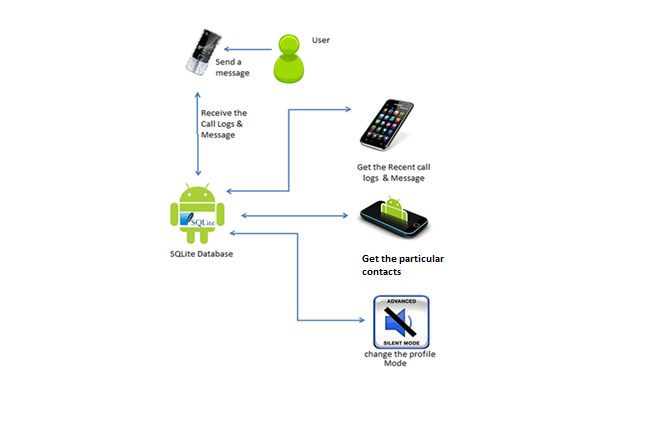


Fig 3.1 Architecture diagram

Call logs is retrieved using content URI. Whenever the person call an other person ,it is registered in the content URI where there is an identification for each calls i.e for Missed call-0, Received call-1and dialled call-2. When the user sends a command , it is passed to content URI and the content URI provides the response to the user. Contacts is retrieved using the content provider which provides the necessary details to the user from one application to another applications . By using SMS Manager, message is retrieved from the android device. All the informations are retrieved only with the proper commands. Once the information is retrieved the user must delete the security key from the other mobile or change the security by using the default settings.

SQLite Database is a Open Source Database. SQLite Database is used for structured data storage . SQLite is a powerful and lightweight relational database engine available to all applications. SQLite is embedded into every Android device. Using an SQLite database in Android does not require a setup procedure or administration of the database.

**3.3 MODULE DESIGN**

**3.3.1 User Authentication Module**

The user authentication module is to check whether the authorized user is logged in. This authentication process is to verify that the given user name and password is valid. Before logged in one time registration is mandatory. In this module a registration page has been created to create username, password and pin number, registration page contains fields of username, password, Pin number and mail ID. When user fill ups all these details and completes his/her registration by clicking register button these details will be registered in SQLite database, which serves as backend. Once the registration is done, the created username and password serves as regular login credentials.

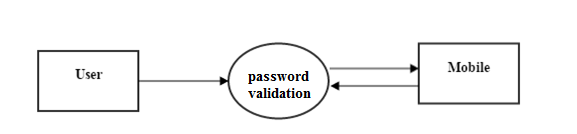


Fig.3.3.1 DFD- User Authentication Module

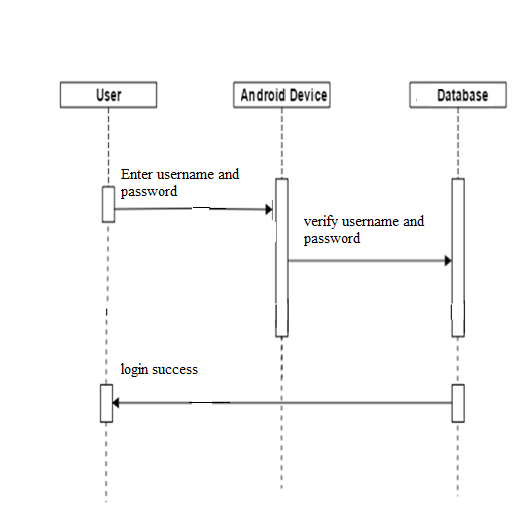


Fig.3.3.1 Sequence Diagram- User Authentication Module

**3.3.2 Customize Commands Module**

The customize commands module is to allow the user to customize his commands by his own. Default commands also available but this customize commands option helps the user to set their own commands for every custom options. User can use both custom and default commands to access the mobile.

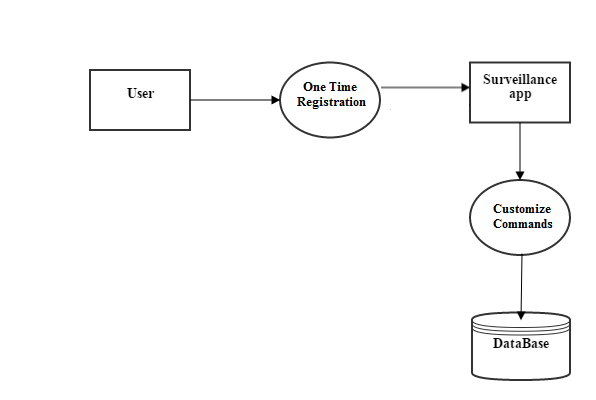


Fig. 3.3.2 DFD- Customize Commands Module

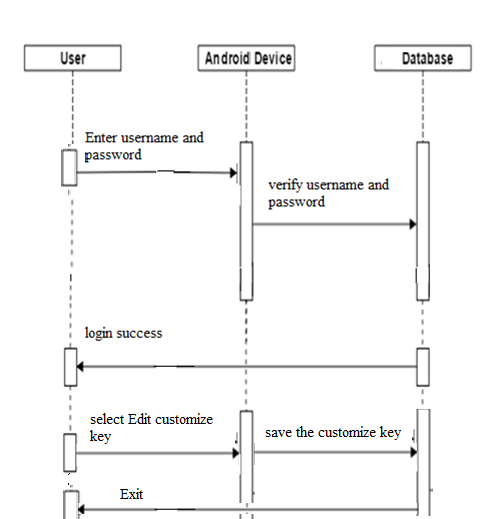


Fig 3.3.2 Sequence Diagram- Customize Commands Module

# 3.3.3 Contact Module

In this module the validation follows with the contact number, if the user needs a contact number in emergency, he/she has to send a message to his/her device which is waiting to reply for the received message format as mentioned below.

* To retrieve contact number **PIN NUMBER- CONTACT-NAME-RES**

If the above conditions satisfy, the broadcast receiver responds a reply to the same number where message got received, for above conditions the contact will be sent as message.

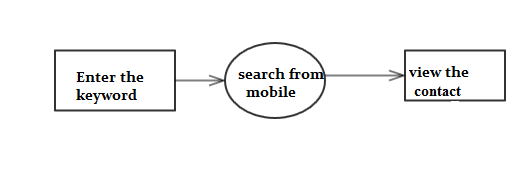
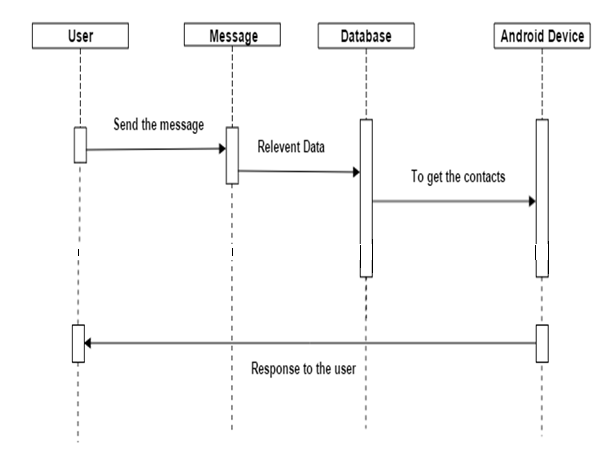


Fig.3.3.3 DFD- Contact Module

Fig.3.3.3 Sequence Diagram-Contact Module

# 3.3.4 Message Module

In this module inbox will be tracked, if the user needs his/her unread SMS form his/her inbox he/she has to send a SMS which satisfies the below condition

* To get unread SMS from Inbox **PIN NUMBER-INBOX-RES**

The SMS will be sent back with the contact number where it got received with the date and time of that SMS.

If the above conditions satisfy, the broadcast receiver responds a reply to the same number where message got received, for above conditions the contact will be sent as message.



Fig 3.3.4 DFD- Message Module

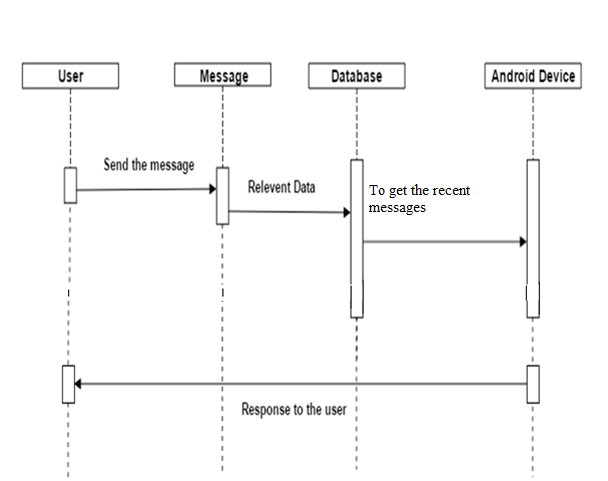


Fig 3.3.4 Sequence Diagram- Message Module

# 3.3.5 Sound Profile Module

In this module user’s mobile sound profile can be changed to vibrate mode, silent mode and Normal mode, for these changes user has to satisfy the below format

* To activate Normal mode **PIN NUMBER-NORMAL-RES**
* To activate Silent mode **PIN NUMBER-SILENT-RES**

If the above conditions satisfy, the broadcast receiver responds a confirmation message reply to the same number where the profile has been change without fail.

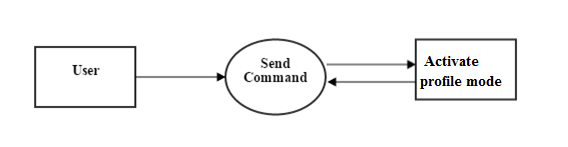
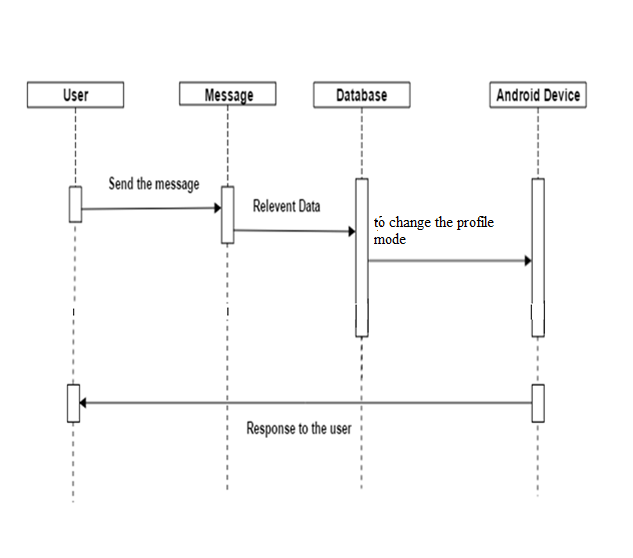


Fig 3.3.5 DFD-Sound Profile

****Fig 3.3.5 Sequence Diagram-Message Module

# 3.3.6 Airplane Mode Module

If the user is willing to deny the missed call and messages to this mobile which is left behind in his/her home, He/she should satisfy the below format

* To activate Airplane Mode **PIN NUMBER-AIRPLANE MODE-RES**

If the flight mode is successfully changed no reply will be received if not invalid statement message will be replied back.

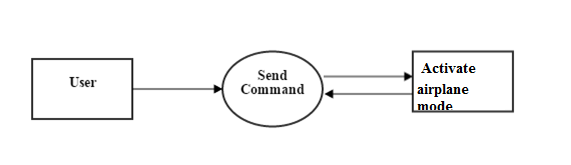
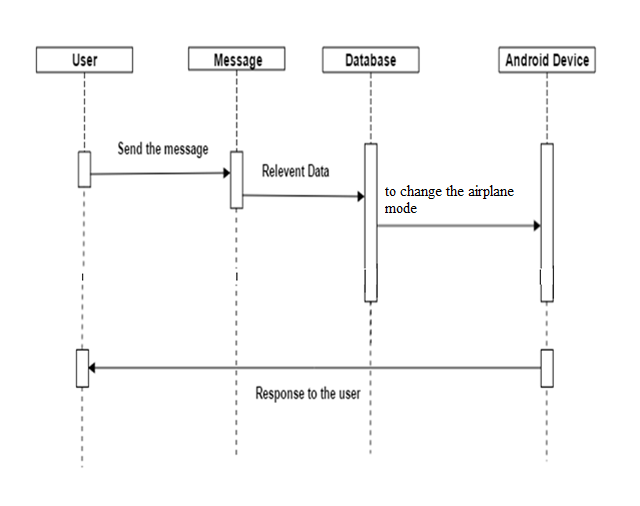


Fig 3.3.6 DFD- Airplane Mode Module

Fig 3.3.6 Sequence Diagram- Airplane Mode Module

# 3.3.7 Call log Module

This module is specially designed for fetching out the call log details such as missed calls, received calls and dialed calls. When the user’s mobile receives a message it goes for validation it has to satisfies the format as listed below,

* Missed call **PIN NUMBER-MISSED CALL-RES**
* Received call **PIN NUMBER-RECEIVED CALL-RES**
* Dialed call **PIN NUMBER- DIALLED CALL-RES**

If the above conditions satisfy, the broadcast receiver responds a reply to the same number where message got received, for above conditions the message will be replied with date and time.

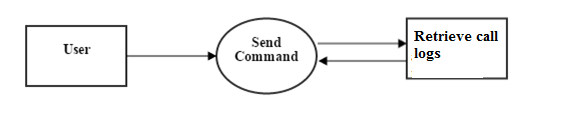


Fig 3.3.7 DFD- Call Logs

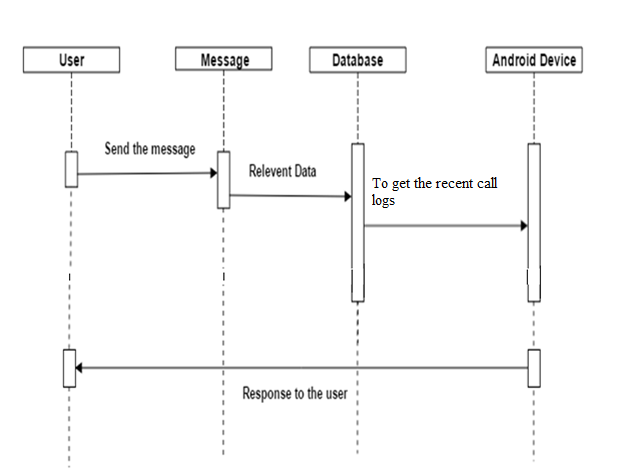


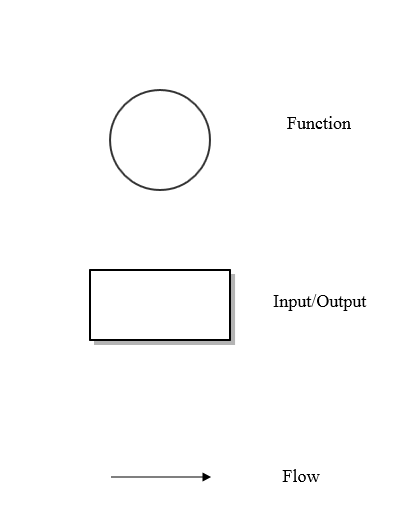
Fig 3.3.7 Sequence Diagram- Call Logs

**3.4 DATA FLOW DIAGRAM**

A Data Flow Diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. Often they are a preliminary step used to Create an overview of the system which can later be elaborated.

A DFD shows what kinds of data will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel.

Symbols Used For Representation:

****

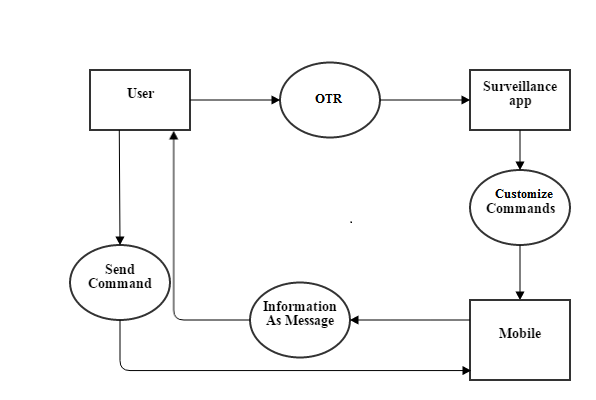
****

Fig 3.4 Dataflow diagram

**3.5 UML Diagram:**

**3.5.1Use case diagram:**

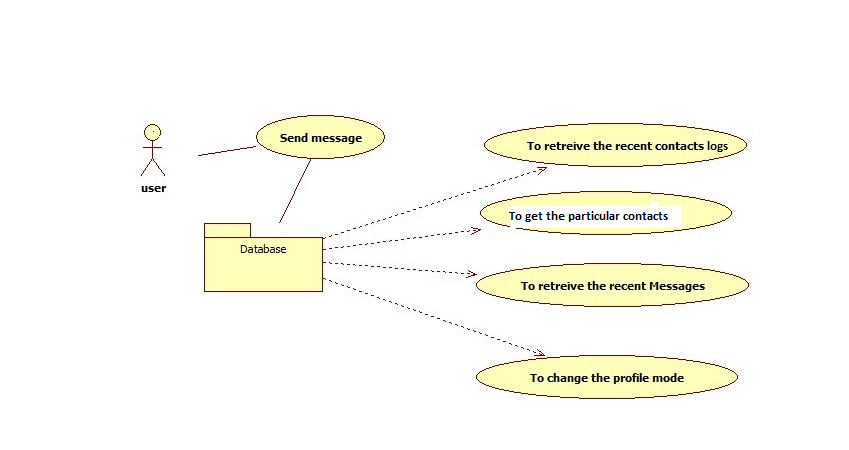
****

Fig 3.5.1 Usecase diagram

**3.5.2 Sequence Diagram**

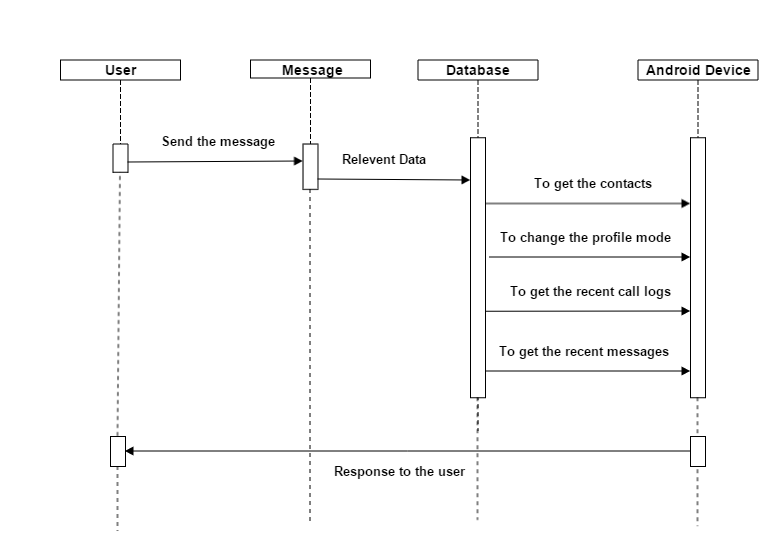
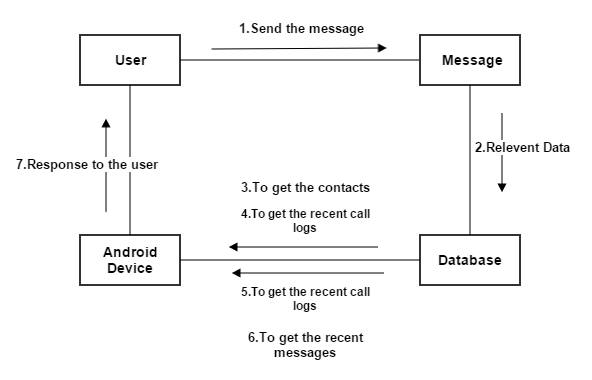
****

Fig 3.5.2 Sequence diagram

**3.5.3 Collaboration Diagram**

**** Fig3.5.3Collaboration diagram

**3.5.4 Activity Diagram**

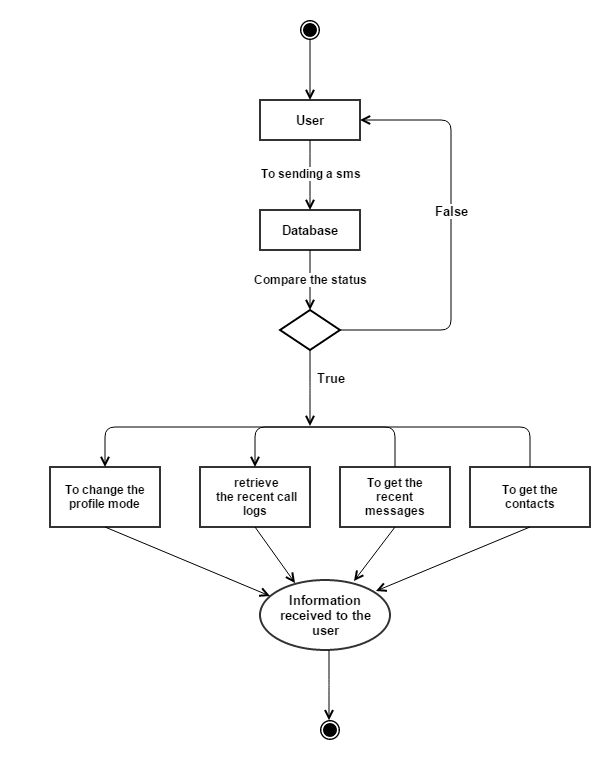
****

FIG 3.5.4 Activity diagram

**3.6 Data Base Design**

A database is a collection of interrelated data stored with minimum redundancy to serve many users quickly and efficiently. The general Objectives of the database design are to make the data access easy, inexpensive and flexible to the user. The data in the system has to be stored and retrieved from database.

Designing the database is the part of system design. Data elements and data structures to be stored have been identified at analysis stage and are structured and put together to design the data storage and retrieval system.

## TABLE 2: Table structure for table user

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Fields** | **Type** | **Null** | **Constraints** | **Description** |
| ***user\_id*** | int(11) | No | Primary Key | Id of the User |
| Username | varchar(20) | No |  | User name of the user |
| Password | varchar(15) | No |  | Password of the user |
| Security Key | varchar(10) | No |  | Security Key which is set by the user |

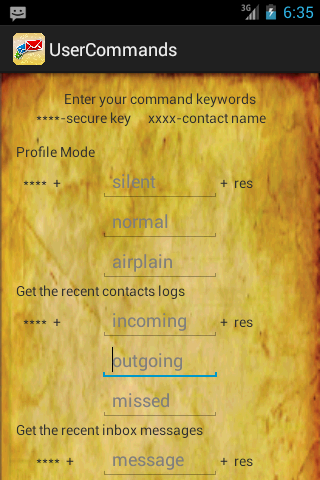
## TABLE 3: Table structure for table command

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Fields** | **Type** | **Null** | **Constraints** | **Description** |
| ***c\_id*** | int(11) | No | Primary Key | Custom Id |
| Silent | varchar(20) | No |  | Command for silent mode |
| Normal | varchar(20) | No |  | Command for normal mode |
| Airplane | varchar(20) | No |  | Command for airplane mode |
| In\_contacts | varchar(20) | No |  | Command for incoming call list |
| Out\_contacts | varchar(20) | No |  | Command for dialed call list |
| Missed\_contacts | varchar(20) | No |  | Command for missed call list |
| Message | varchar(20) | No |  | Command for retrieve message |

**3.7 INPUT DESIGN**

The input of a system can be defined as the information that is provided to the system. This is used for future processing by the system to obtain meaningful information, which helps in decision-making. Input design is the process of converting user-oriented inputs to a computer-based format.

Input is a part of overall system design, which requires special attention. Inaccurate input data are the most common cause of errors in error processing. Input design can control errors entered by users. Entered data have to be checked for their accuracy and direction of errors. Appropriate error message have to be displayed. When an invalid data is entered, the user should not be allowed to type that data.

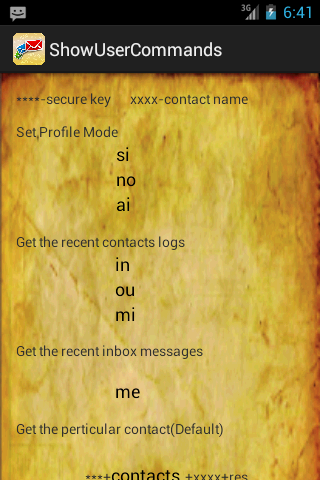


Screenshot 3.7 Input Design

**3.8 OUTPUT DESIGN**

The computer output is the most important and direct source of information to the user. Efficient and intelligible output design improves the system’s relationship with the user and helps in decision making.

Output design was studied going actively during the study phase. The objective of the output design is defined the contents and format of all documents and reports in an attractive and useful format.



Screenshot 3.8 Output Design

**REQUIREMENT SPECIFICATION**

**4.1 SYSTEM SPECIFICATIONS**

**4.1.1 Hardware Requirements:**

Processor : Pentium P4

Motherboard : Genuine Intel

RAM : Min 1 GB

Hard Disk : 80 GB

Mobile : Android Based Smart phones

**4.1.2 Software Requirements:**

Operating system : Windows XP

Technology Used : Android 2.2

IDE : Eclipse Indigo

Emulators : AVD

Plug-in : ADT plug-in

Database : SQLite

Tools used : Android SDK

**4.2 SOFTWARE DESCRIPTION**

Software can be thought of as the variable part of a computer and hardware the invariable part. Software is often divided into [application](http://searchsoftwarequality.techtarget.com/definition/application) software (programs that do work users are directly interested in) and [system](http://searchwinit.techtarget.com/definition/system) software (which includes [operating system](http://searchcio-midmarket.techtarget.com/definition/operating-system)s and any program that supports application software). The term [middleware](http://searchsoa.techtarget.com/definition/middleware) is sometimes used to describe programming that mediates between application and system software or between two different kinds of application software (for example, sending a remote work request from an application in a computer that has one kind of operating system to an application in a computer with a different operating system).

**4.2.1 FRONT END**

**4.2.1.1 Android**

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. Android is a software platform and operating system for mobile devices based on the Linux operating system and developed by Google and the Open Handset Alliance. It allows developers to write managed code in a Java-like language that utilizes Google-developed Java libraries, but does not support programs developed in native code. The unveiling of the Android platform on 5 November 2007 was announced with the founding of the Open Handset Alliance, a consortium of 34 hardware, software and telecom companies devoted to advancing open standards for mobile devices. When released in 2008, most of the Android platform will be made available under the apache free-software and open-source license.

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. Android is a software platform and operating system for mobile devices based on the Linux operating system and developed by Google and the Open Handset Alliance. It allows developers to write managed code in a Java-like language that utilizes Google-developed Java libraries, but does not support programs developed in native code.

Android  is a powerful operating system supporting a large number of applications in [smart phones](http://www.engineersgarage.com/articles/smart-phones). these applications make life more comfortable and advanced for the users. hardwares that support android are mainly based on [ARM architecture](http://www.engineersgarage.com/articles/arm-advanced-risc-machines-processors) platform.

The unveiling of the Android platform on 5 November 2007 was announced with the founding of the Open Handset Alliance, a consortium of 34 hardware, software and telecom companies devoted to advancing open standards for mobile devices. When released in 2008, most of the Android platform will be made available under the Apache free-software and open-source license.

Android comes with an Android market which is an online software store. It was developed by Google. It allows Android users to select, and download applications developed by third party developers and use them. There are around 2.0 lack+ games, application and widgets available on the market for users.

Android applications are written in java programming language. Android is available as open source for developers to develop applications which can be further used for selling in android market. There are around 200000 applications developed for android with over 3 billion+ downloads. Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model.

**4.2.1.2 Applications**

Android applications are composed of one or more application components (activities, services, content providers, and broadcast receivers).  Each component performs a different role in the overall application behavior, and each one can be activated individually (even by other applications). The manifest file must declare all components in the application and should also declare all application requirements, such as the minimum version of Android required and any hardware configurations required.Non-code application resources (images, strings, layout files, etc.) should include alternatives for different device configurations (such as different strings for different languages).

**4.2.2. BACK END**

**4.2.2.1 SQLite Database**

SQLite is an Open Source database. SQLite supports standard relational database features like SQL syntax, transactions and prepared statements. The database requires limited memory at runtime (approx. 250 KByte) which makes it a good candidate from being embedded into other runtimes. SQLite is embedded into every Android device. Using an SQLite database in Android does not require a setup procedure or administration of the database. It is based on function calls and single file, where all definitions, tables and data are stored. This simple design is more than suitable for a platform such as Android.

**4.2.2.2 Handset Layouts**

The platform is adaptable to both larger, VGA, 2D graphics library, 3D graphics library based on OpenGL ES 1.0 specifications, traditional smart phone layouts. An underlying 2D graphics engine is also included. Surface Manager manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications.  
**4.2.2.3 Data Storage**

SQLite is used for structured data storage .SQLite is a powerful and lightweight relational database engine available to all applications.  
**4.2.2.4Connectivity**   
 Android supports a wide variety of connectivity technologies including GSM, CDMA, Bluetooth, EDGE, EVDO, 3G and Wi-Fi.

**4.2.2.5 Java Virtual Machine**

Software written in Java can be compiled into Dalvik bytecodes and executed in the Dalvik virtual machine, which is a specialized VM implementation designed for mobile device use, although not technically a standard Java Virtual Machine. all programming language compilers convert the source code to machine code. Same job done by Java Compiler to run a Java program, but the difference is that Java compiler convert the source code into Intermediate code is called as bytecode. This machine is called the Java Virtual machine and it exits only inside the computer memory.

**4.2.2.6 Media Support**

Android will support advanced audio/video/still media formats such as MPEG-4, H.264, MP3, and AAC, AMR, JPEG, PNG, GIF.   
**4.2.2.7 Additional Hardware Support**

Android is fully capable of utilizing video/still cameras, touchscreens, GPS, compasses, accelerometers, and accelerated 3D graphics.

**4.2.2.8 Development Environment**

Includes a device emulator, tools for debugging, memory and performance profiling, a plugin for the Eclipse IDE. There are a number of hardware dependent features, for instance, a huge media and connections support, GPS, improved support for Camera and simply GSM telephony. A great work was done for the developers to start work with Android using device emulator, tools for debugging and plug-in for Eclipse IDE.

**SYSTEM IMPLEMENTATION**

A software application in general is implemented after navigating the complete life cycle method of a project. Various life cycle processes such as requirement analysis, design phase, verification, testing and finally followed by the implementation phase result in a successful project management. System implementation is an important stage of theoretical design is turned into practical system.

**5.1 IMPLEMENTATION PROCEDURE**

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective. The implementation stage involves careful planning, investigation of the existing system and it’s constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

Each program is tested individually at the time of development using the data and has verified that this program linked together in the way specified in the programs specification, the computer system and its environment is tested to the satisfaction of the user. The system that has been developed is accepted and proved to be satisfactory for the user and so the system is going to be implemented very soon. A simple operating procedure is included so that the user can understand the different functions clearly and quickly. The final stage is to document the entire system which provides components and the operating procedures of the system.

**5.2 ALGORITHM**

Step 1: To register the in application register page (One time registration) and give

the secure key to handle your application process.

Step2: While you register choose default key or customize key.

Step 3: If you choose customize key fill all the key value to register and it

automatically start the service.

Step 4: If you need to modify the settings go to login page and select the

customization as your need.

Step 5: Give the secure key and your customizable key and res (default key) to

access the prior information from your phone.

Step 6: You can enable the service otherwise disable the service as your need.

Step 7: If the service is enabled it provides the required information using

**Classical Partitioning Algorithm** which compares the given command

with the application service by separating the commands into parts.

Step 8: If the service is disabled the conversation must be deleted.

**5.3 SOURCE CODE**

**User authentication code**

**Loginactivity.java**

package com.example.readmsg;

import android.os.Bundle;

import android.app.Activity;

import android.content.Intent;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.view.Menu;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;

public class LoginActivity extends Activity {

SQLiteDatabase dbs;

EditText e\_username;

EditText e\_password;

String s\_password="",s\_username="";

Button btn;

String ss\_username="", ss\_password="", s\_securepass="";

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_login);

dbs = openOrCreateDatabase("datacollect.db", MODE\_PRIVATE, null);

String sql\_create = "create table if not exists login(\_id integer primary key autoincrement, username varchar(120),password varchar(120),secret varchar(120))";

dbs.execSQL(sql\_create);

e\_username = (EditText)findViewById(R.id.loginusername);

e\_password = (EditText)findViewById(R.id.loginpassword);

btn=(Button)findViewById(R.id.login);

btn.setOnClickListener(new OnClickListener()

{

public void onClick(View v) {

ss\_username=e\_username.getText().toString();

ss\_password=e\_password.getText().toString();

if(ss\_username.equals("")||ss\_username==null || ss\_password.equals("")||ss\_password==null)

{

Toast.makeText(getApplicationContext(), "Fill all above details to proceed", Toast.LENGTH\_LONG).show();

}

else{

String s\_select = "select \* from login";

Cursor mCursor = dbs.rawQuery(s\_select, null);

if(mCursor!=null)

{

while(mCursor.moveToNext())

{

s\_username=mCursor.getString(mCursor.getColumnIndex("username"));

s\_securepass=mCursor.getString(mCursor.getColumnIndex("password"));

s\_password = mCursor.getString(mCursor.getColumnIndex("secret"));

if(ss\_username.equals(s\_username) && ss\_password.equals(s\_securepass))

{

Toast.makeText(getApplicationContext(), "Login success", Toast.LENGTH\_LONG).show();

Intent intent=new Intent(getApplicationContext(),SecureActivity.class);

startActivity(intent);

finish();

}

else{

Toast.makeText(getApplicationContext(), "Enter the valid username and password", Toast.LENGTH\_LONG).show();

}

}

}

}

}

});

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.login, menu);

return true;

}

}

**Customize commands module**

**Mainactivity.java**

package com.example.readmsg;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import android.os.Bundle;

import android.app.Activity;

import android.view.Menu;

import android.view.View;

import android.widget.Button;

import android.widget.CheckBox;

import android.widget.EditText;

import android.widget.RadioButton;

import android.widget.RadioGroup;

import android.widget.Toast;

import android.content.Intent;

import android.database.sqlite.SQLiteDatabase;

import android.view.View.OnClickListener;

public class MainActivity extends Activity implements OnClickListener {

Button btn\_save;

EditText e\_firstname,e\_lastname,e\_username,e\_password,e\_email,secret;

CheckBox cbx,cbxs;

String $username;

String $password;

int id;

RadioGroup radioSexGroup;

private RadioButton radioSexButton;

String getusername,getpassword;

SQLiteDatabase dbs;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

dbs = openOrCreateDatabase("datacollect.db", MODE\_PRIVATE, null);

String sql\_create = "create table if not exists login(\_id integer primary key autoincrement, username varchar(120),password varchar(120),secret varchar(120))";

dbs.execSQL(sql\_create);

radioSexGroup = (RadioGroup) findViewById(R.id.goodbutton);

e\_username = (EditText)findViewById(R.id.username);

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

secret=(EditText)findViewById(R.id.secure);

e\_email=(EditText)findViewById(R.id.email);

e\_firstname=(EditText)findViewById(R.id.firstname);

e\_lastname=(EditText)findViewById(R.id.lastname);

btn\_save = (Button)findViewById(R.id.login);

btn\_save.setOnClickListener(this);

}

private boolean isValidEmail(String email) {

String EMAIL\_PATTERN = "^[\_A-Za-z0-9-\\+]+(\\.[\_A-Za-z0-9-]+)\*@"

+ "[A-Za-z0-9-]+(\\.[A-Za-z0-9]+)\*(\\.[A-Za-z]{2,})$";

Pattern pattern = Pattern.compile(EMAIL\_PATTERN);

Matcher matcher = pattern.matcher(email);

return matcher.matches();

}

// validating password with retype password

private boolean isValidPassword(String pass) {

if (pass != null && pass.length() >= 2) {

return true;

}

return false;

}

public void onClick(View v)

{

switch(v.getId())

{

case R.id.login:

String s\_username = e\_username.getText().toString();

String s\_password = e\_password.getText().toString();

String s\_email=e\_email.getText().toString();

String s\_securekey=secret.getText().toString();

int selectedId = radioSexGroup.getCheckedRadioButtonId();

// find the radiobutton by returned id

radioSexButton = (RadioButton) findViewById(selectedId);

String command = radioSexButton.getText().toString().trim();

if(s\_password.equals("") || s\_password==null || s\_username.equals("") || s\_username==null

&& s\_email.equals("")||s\_email==null || s\_securekey.equals("")||s\_securekey==null)

{

Toast.makeText(getApplicationContext(), "Fill all above details to proceed", Toast.LENGTH\_LONG).show();

}

else if(!isValidEmail(s\_email)){

Toast.makeText(getApplicationContext(), "Enter the valid email", Toast.LENGTH\_LONG).show();

}

else if(!isValidPassword(s\_password)){

Toast.makeText(getApplicationContext(), "Valid password length greater than Two", Toast.LENGTH\_LONG).show();

}

else{

String inse\_query = "insert into login(username,password,secret) values('"+s\_username+"','"+s\_password+"','"+s\_securekey+"')";

dbs.execSQL(inse\_query);

Toast.makeText(getApplicationContext(), "Success", Toast.LENGTH\_SHORT).show();

if(command.equals("Default Commands"))

{

Intent intent=new Intent();

intent.setClassName("com.example.readmsg","com.example.readmsg.CommandLine");

intent.setFlags(intent.FLAG\_ACTIVITY\_NEW\_TASK);

startActivity(intent);

Toast.makeText(getApplicationContext(), "Use default command keywords", Toast.LENGTH\_LONG).show();

finish();

}

else if(command.equals("User Commands"))

{

Intent intents=new Intent();

intents.setClassName("com.example.readmsg","com.example.readmsg.UserCommands");

intents.setFlags(intents.FLAG\_ACTIVITY\_NEW\_TASK);

startActivity(intents);

// Toast.makeText(getApplicationContext(), "Success", Toast.LENGTH\_SHORT).show();

Toast.makeText(getApplicationContext(), "Enter the following commands keywords", Toast.LENGTH\_LONG).show();

finish();

}

else

{

Toast.makeText(getApplicationContext(), "Sorry cann't find", Toast.LENGTH\_SHORT).show();

}

}

break;

}

}

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.main, menu);

return true;

}

}

**Contacts module**

**Contacts.java**

package com.example.readmsg;

import android.os.Bundle;

import android.provider.ContactsContract;

import android.app.Activity;

import android.content.ContentResolver;

import android.content.Intent;

import android.database.Cursor;

import android.telephony.SmsManager;

import android.view.Menu;

import android.widget.TextView;

import android.widget.Toast;

public class Contacts extends Activity {

TextView text\_contact;

String contact="";

String para="",para1="";

String e\_con="";

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_contacts);

text\_contact = (TextView)findViewById(R.id.text\_cont);

Intent extras=getIntent();

para=extras.getStringExtra("str");

para1= extras.getStringExtra("str1");

Toast.makeText(getApplicationContext(), "phoneNumber"+para1,Toast.LENGTH\_LONG).show();

displayContacts();

}

public void displayContacts(){

StringBuffer sb=new StringBuffer();

ContentResolver cr=getContentResolver();

Cursor cur=cr.query(ContactsContract.Contacts.CONTENT\_URI, null, null, null, null);

//String whereName = ContactsContract.Data.MIMETYPE + " = ?";

// String[] whereNameParams = new String[] { ContactsContract.CommonDataKinds.StructuredName.CONTENT\_ITEM\_TYPE };

// Cursor cur = cr.query(ContactsContract.Data.CONTENT\_URI, null, whereName, whereNameParams, ContactsContract.CommonDataKinds.StructuredName.GIVEN\_NAME);

if (cur.getCount() > 0)

{

while (cur.moveToNext())

{

String id = cur.getString(cur.getColumnIndex(ContactsContract.Contacts.\_ID));

String name = cur.getString(cur.getColumnIndex(ContactsContract.Contacts.DISPLAY\_NAME));

if (Integer.parseInt(cur.getString(

cur.getColumnIndex(ContactsContract.Contacts.HAS\_PHONE\_NUMBER))) > 0) {

Cursor pCur = cr.query(

ContactsContract.CommonDataKinds.Phone.CONTENT\_URI,

null,

ContactsContract.CommonDataKinds.Phone.CONTACT\_ID +" = ?",

new String[]{id}, null);

while (pCur.moveToNext()) {

String phoneNo = pCur.getString(pCur.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER));

//Toast.makeText(Contacts.this, "Name: " + name + ", Phone No: " + phoneNo, Toast.LENGTH\_LONG).show();

if(name.contentEquals(para)){

sb.append("\n"+name+" : "+phoneNo);

}

}

// text\_contact.setText(sb);

// pCur.close();

}

}

try {

SmsManager smsManager = SmsManager.getDefault();

smsManager.sendTextMessage(para1, null, sb.toString().trim(), null, null);

Toast.makeText(getApplicationContext(), "SMS Sent!",

Toast.LENGTH\_LONG).show();

} catch (Exception e) {

SmsManager smsManager = SmsManager.getDefault();

smsManager.sendTextMessage(para1, null,"Irrelevent contact Matching to the DB", null, null);

}

}

finish();

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.contacts, menu);

return true;

}}

**Message module**

**Incomingsms.java**

package com.example.readmsg;

import android.app.Activity;

import android.app.KeyguardManager;

import android.app.KeyguardManager.KeyguardLock;

import android.content.BroadcastReceiver;

import android.content.Context;

import android.content.Intent;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.media.AudioManager;

import android.os.Build;

import android.os.Bundle;

import android.os.PowerManager;

import android.os.PowerManager.WakeLock;

import android.provider.Settings;

import android.telephony.SmsManager;

import android.telephony.SmsMessage;

import android.util.Log;

import android.widget.Toast;

public class IncomingSms extends BroadcastReceiver{

AudioManager am;

PowerManager powermanager;

WakeLock wakeLock;

public static String body\_content;

String part1="";

String part2="",part3="",part4="";

String messageContent1="",messageContent2="",messageContent3="";

@SuppressWarnings("deprecation")

String s\_password = "";

SQLiteDatabase dbs = null;

String silent1="",normal1="",airplain1="",incoming1="",outgoing1="",missed1="",contact1="",message1="";

public void onReceive(Context context, Intent intent) {

final Bundle bundle=intent.getExtras();

String senderNum="";

int duration=Toast.LENGTH\_LONG;

try{

dbs = context.openOrCreateDatabase("datacollect.db",Context.MODE\_PRIVATE, null);

String st="create table if not exists getuser(\_id integer primary key autoincrement,silent varchar(20),normal varchar(20),airplain varchar(20),incoming varchar(20),outgoing varchar(20),missed varchar(20),contacts varchar(20),message varchar(20))";

dbs.execSQL(st);

dbs = context.openOrCreateDatabase("datacollect.db",Context.MODE\_PRIVATE, null);

String sql\_create = "create table if not exists login(\_id integer primary key autoincrement, username varchar(120),password varchar(120),secret varchar(120))";

dbs.execSQL(sql\_create);

String selects = "select \* from getuser";

Cursor mmCursor = dbs.rawQuery(selects, null);

if(mmCursor!=null)

{

while(mmCursor.moveToNext())

{

silent1 = mmCursor.getString(mmCursor.getColumnIndex("silent"));

normal1 = mmCursor.getString(mmCursor.getColumnIndex("normal"));

airplain1 = mmCursor.getString(mmCursor.getColumnIndex("airplain"));

incoming1 = mmCursor.getString(mmCursor.getColumnIndex("incoming"));

outgoing1 = mmCursor.getString(mmCursor.getColumnIndex("outgoing"));

missed1 = mmCursor.getString(mmCursor.getColumnIndex("missed"));

message1=mmCursor.getString(mmCursor.getColumnIndex("missed"));

contact1 = mmCursor.getString(mmCursor.getColumnIndex("contacts"));

}

}

}

catch(Exception e){

e.printStackTrace();

}

try

{

if(bundle!=null){

final Object[] obj=(Object[])bundle.get("pdus");

for(int i=0; i<obj.length;i++){

SmsMessage currentMessage=SmsMessage.createFromPdu((byte[]) obj[i]);

String phoneNumber=currentMessage.getDisplayOriginatingAddress();

senderNum=phoneNumber;

String message=currentMessage.getDisplayMessageBody();

Log.i("SmsReceiver","SenderNum"+senderNum+"message"+message);

int duration1=Toast.LENGTH\_LONG;

body\_content = message.toString().trim();

dbs = context.openOrCreateDatabase("datacollect.db",Context.MODE\_PRIVATE, null);

String s\_select = "select \* from login";

Cursor mCursor = dbs.rawQuery(s\_select, null);

if(mCursor!=null)

{

while(mCursor.moveToNext())

{

s\_password = mCursor.getString(mCursor.getColumnIndex("secret"));

}

}

StringBuilder sb=new StringBuilder().append(s\_password).append("-contacts");

if(body\_content.endsWith("res")) {

if(body\_content.startsWith(s\_password)&& body\_content.endsWith("res")){

if(body\_content.startsWith(sb.toString()))

{

String string1 =message;

String[] parts = string1.split("-");

part1 = parts[0];

part2 = parts[1];

part3 = parts[2];

part4 = parts[3];

}

else if(body\_content.startsWith(s\_password)){

String string=message;

String[] parts=string.split("-");

messageContent1=parts[0];

messageContent2=parts[1];

messageContent3=parts[2];

}

//

//

// else

// {

// Intent intent4=new Intent();

// intent4.setClassName("com.example.readmsg", "com.example.readmsg.SmsResponse");

// intent4.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK);

// intent4.putExtra("str1",senderNum);

// context.startActivity(intent4);

// }

}

if(part2.equals("contacts")||messageContent2.equals(contact1)){

Intent intent2=new Intent();

intent2.setClassName("com.example.readmsg","com.example.readmsg.Contacts");

intent2.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK);

intent2.putExtra("str",part3);

intent2.putExtra("str1",senderNum);

context.startActivity(intent2);

}

else if(messageContent2.equals("silent")||messageContent2.equals(silent1))

{

am= (AudioManager)context.getSystemService(Context.AUDIO\_SERVICE);

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

Toast.makeText(context,"SILENT profile activated ",Toast.LENGTH\_LONG).show();

SmsManager sm = SmsManager.getDefault();

sm.sendTextMessage(senderNum, null,"Silent Mode Activated", null,

null);

}

else if(messageContent2.equals("airplane mode")||messageContent2.equals(airplain1))

{

Settings.System.putInt(context.getContentResolver(),Settings.System.AIRPLANE\_MODE\_ON, 1);//Turning ON Airplane mode.

Intent intent4 = new Intent(Intent.ACTION\_AIRPLANE\_MODE\_CHANGED);//creating intent and Specifying action for AIRPLANE mode.

intent4.putExtra("state", true);////indicate the "state" of airplane mode is changed to ON

context.sendBroadcast(intent4);//Broadcasting and Intent

Toast.makeText(context,"airplane mode is on ",Toast.LENGTH\_LONG).show();

SmsManager sm = SmsManager.getDefault();

sm.sendTextMessage(senderNum, null,"airplane Mode Activated", null,

null);

}

else if(messageContent2.equals("normal")||messageContent2.equals(normal1))

{

am= (AudioManager)context.getSystemService(Context.AUDIO\_SERVICE);

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

Toast.makeText(context,"NORMAL profile activated ",Toast.LENGTH\_LONG).show();

SmsManager sm = SmsManager.getDefault();

sm.sendTextMessage(senderNum, null,"Normal Mode Activated", null,

null);

}

else if(messageContent2.equals("incoming logs")||messageContent2.equals(incoming1))

{

Intent intent1=new Intent();

intent1.setClassName("com.example.readmsg", "com.example.readmsg.IncommingLogs");

intent1.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK);

intent1.putExtra("str","Incoming");

intent1.putExtra("str1",senderNum);

context.startActivity(intent1);

}

else if(messageContent2.equals("outgoing logs")||messageContent2.equals(outgoing1))

{

Intent intent1=new Intent();

intent1.setClassName("com.example.readmsg", "com.example.readmsg.IncommingLogs");

intent1.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK);

intent1.putExtra("str","Outgoing");

intent1.putExtra("str1",senderNum);

context.startActivity(intent1);

}

else if(messageContent2.equals("missed logs")||messageContent2.equals(missed1))

{

Intent intent1=new Intent();

intent1.setClassName("com.example.readmsg", "com.example.readmsg.IncommingLogs");

intent1.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK);

intent1.putExtra("str","Missed");

intent1.putExtra("str1",senderNum);

context.startActivity(intent1);

}

else if(messageContent2.equals("inbox message")||messageContent2.equals(message1))

{

Intent intent1=new Intent();

intent1.setClassName("com.example.readmsg", "com.example.readmsg.ReadSmsActivity");

intent1.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK);

intent1.putExtra("str2",senderNum);

context.startActivity(intent1);

}

else

{

Intent intent3=new Intent();

intent3.setClassName("com.example.readmsg","com.example.readmsg.SmsResponse");

intent3.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK);

intent3.putExtra("str1",senderNum);

context.startActivity(intent3);

}

}

}

}

}catch(Exception e){

Log.e("SmsReceiver","Exception smsReceiver"+e);

}

}

}

**Readsmsactivity.java**

package com.example.readmsg;

import java.io.BufferedReader;

import android.net.Uri;

import android.os.Bundle;

import android.app.Activity;

import android.content.Intent;

import android.database.Cursor;

import android.telephony.SmsManager;

import android.util.Log;

import android.view.Menu;

import android.widget.TextView;

import android.widget.Toast;

public class ReadSmsActivity extends Activity {

TextView text\_readsms;

String date;

static String msgData = "";

String msgData1="";

String para;

BufferedReader in;

String read;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_read\_sms);

Intent extras = getIntent();

para= extras.getStringExtra("str2");

Toast.makeText(getApplicationContext(), "phoneNumber"+para,Toast.LENGTH\_LONG).show();

int count =0;

Cursor cursor = this.getContentResolver().query(Uri.parse("content://sms/inbox"), null, null, null, null);

cursor.moveToFirst();

do{

if(count<5)

{

msgData +=cursor.getString(cursor.getColumnIndexOrThrow("address"))+"\n"+":"+cursor.getString(cursor.getColumnIndex("body"))+"\n";

msgData +=("~");

count++;

}

else

{

msgData1 +="From "+cursor.getString(cursor.getColumnIndexOrThrow("address"))+"\n"+"Message Body "+cursor.getString(cursor.getColumnIndex("body"))+

"\n"+"Date "+cursor.getString(cursor.getColumnIndexOrThrow("date"))+"\n";

}

}while(cursor.moveToNext());

String string =msgData;

String[] parts = string.split("~");

for(int i =0;i<parts.length && i<=4;i++)

{

try {

SmsManager smsManager = SmsManager.getDefault();

smsManager.sendTextMessage(para, null,parts[i].toString(), null, null);

Toast.makeText(getApplicationContext(), "SMS Sent!",

Toast.LENGTH\_LONG).show();

} catch (Exception e) {

Toast.makeText(getApplicationContext(),

"SMS faild, please try again later!",

Toast.LENGTH\_LONG).show();

e.printStackTrace();

}

}

finish();

}

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.read\_sms, menu);

return true;

}

}

**Call logs module**

**Incominglogs.java**

package com.example.readmsg;

import java.sql.Date;

import android.app.Activity;

import android.content.Intent;

import android.database.Cursor;

import android.os.Bundle;

import android.provider.CallLog;

import android.telephony.SmsManager;

import android.widget.TextView;

import android.widget.Toast;

public class IncommingLogs extends Activity {

TextView textView = null;

String contact="";

String contact1="";

String para="",para1="";

String first="";

int i=0;

boolean contend = false;

static boolean ring = false;

static boolean callReceived = false;

StringBuffer sb=new StringBuffer();

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_incomming\_logs);

Intent extras = getIntent();

para= extras.getStringExtra("str");

textView = (TextView) findViewById(R.id.textview\_call);

para1= extras.getStringExtra("str1");

Toast.makeText(getApplicationContext(), "phoneNumber"+para1,Toast.LENGTH\_LONG).show();

if (para.equals("Incoming")){

getCallDetails();

}

if (para.equals("Outgoing")){

getCallDetails1();

}

if (para.equals("Missed")){

getCallDetails2();

}

finish();

}

private void getCallDetails() {

StringBuffer sb = new StringBuffer();

Cursor managedCursor = managedQuery(CallLog.Calls.CONTENT\_URI, null,null, null, null);

int number = managedCursor.getColumnIndex(CallLog.Calls.NUMBER);

int type = managedCursor.getColumnIndex(CallLog.Calls.TYPE);

int date = managedCursor.getColumnIndex(CallLog.Calls.DATE);

// int duration = managedCursor.getColumnIndex(CallLog.Calls.DURATION);

// sb.append("Call Log :");

int count=0;

managedCursor.moveToLast();

do{

String phNumber = managedCursor.getString(number);

String callType = managedCursor.getString(type);

String callDate = managedCursor.getString(date);

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

// String callDuration = managedCursor.getString(duration);

if (callType.contentEquals("1")){

if(count<5){

sb.append("\nPh" + phNumber+"D"+ callDayTime);

count++;

}

else

{

contact1 +="\nPhNo" + phNumber + "Date" + callDayTime;

}

}

}while(managedCursor.moveToPrevious());

textView.setText(sb);

// String e\_con = textView.getText().toString().trim();

SmsManager sm = SmsManager.getDefault();

sm.sendTextMessage(para1, null,sb.toString(), null,

null);

}

private void getCallDetails1() {

StringBuffer sb = new StringBuffer();

Cursor managedCursor = managedQuery(CallLog.Calls.CONTENT\_URI, null,

null, null, null);

int number = managedCursor.getColumnIndex(CallLog.Calls.NUMBER);

int type = managedCursor.getColumnIndex(CallLog.Calls.TYPE);

int date = managedCursor.getColumnIndex(CallLog.Calls.DATE);

// int duration = managedCursor.getColumnIndex(CallLog.Calls.DURATION);

// sb.append("Call Log :");

int count=0;

managedCursor.moveToLast();

do{

String phNumber = managedCursor.getString(number);

String callType = managedCursor.getString(type);

String callDate = managedCursor.getString(date);

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

// String callDuration = managedCursor.getString(duration);

if (callType.contentEquals("2")){

if(count<5){

sb.append("\nPh" + phNumber+"D"+ callDayTime);

count++;

}

else

{

contact1 +="\nPhNo" + phNumber + " \tDate" + callDayTime;

}

}

}while(managedCursor.moveToPrevious());

textView.setText(sb);

// String e\_con = textView.getText().toString().trim();

SmsManager sm = SmsManager.getDefault();

sm.sendTextMessage(para1, null,sb.toString(), null,

null);

}

private void getCallDetails2() {

StringBuffer sb = new StringBuffer();

Cursor managedCursor = managedQuery(CallLog.Calls.CONTENT\_URI, null,null, null, null);

int number = managedCursor.getColumnIndex(CallLog.Calls.NUMBER);

int type = managedCursor.getColumnIndex(CallLog.Calls.TYPE);

int date = managedCursor.getColumnIndex(CallLog.Calls.DATE);

// int duration = managedCursor.getColumnIndex(CallLog.Calls.DURATION);

// sb.append("Call Log :");

int count=0;

managedCursor.moveToLast();

do{

String phNumber = managedCursor.getString(number);

String callType = managedCursor.getString(type);

String callDate = managedCursor.getString(date);

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

// String callDuration = managedCursor.getString(duration);

if (callType.contentEquals("3")){

if(count<5){

sb.append("\nPh" + phNumber+"D"+ callDayTime);

count++;

}

else

{

contact1 +="\nPhNo" + phNumber + " \tDate" + callDayTime;

}

}

}while(managedCursor.moveToPrevious());

textView.setText(sb);

// String e\_con = textView.getText().toString().trim();

SmsManager sm = SmsManager.getDefault();

sm.sendTextMessage(para1, null,sb.toString(), null,

null);

}

}

**Smsresponse.java**

package com.example.readmsg;

import android.os.Bundle;

import android.app.Activity;

import android.content.Intent;

import android.telephony.SmsManager;

import android.view.Menu;

public class SmsResponse extends Activity {

String para="";

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_sms\_response);

Intent extras=getIntent();

para=extras.getStringExtra("str1");

SmsManager sm = SmsManager.getDefault();

sm.sendTextMessage(para, null,"Invalid Statement", null,null);

finish();

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

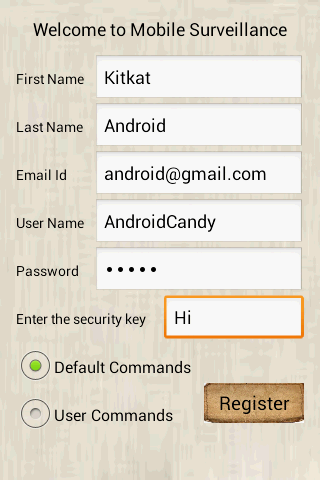
getMenuInflater().inflate(R.menu.sms\_response, menu);

return true;

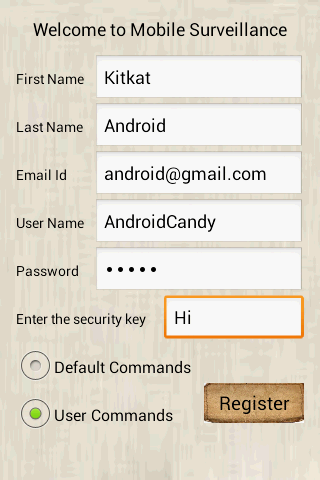
}

}

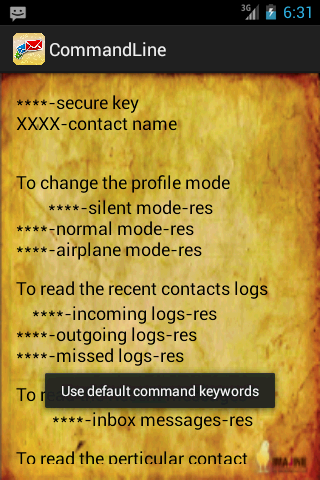
**5.4 SCREENSHOTS**



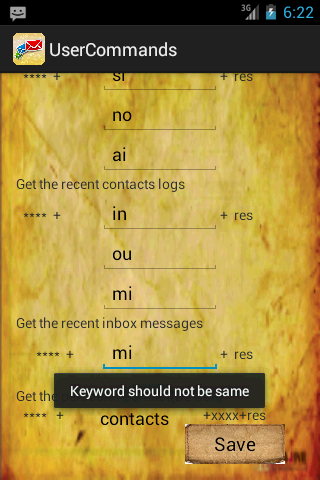
Screenshot:1 Registration page with Default commands



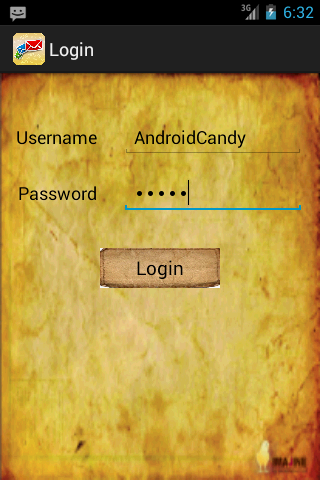
Screenshot: 2 Registration page with User Commands



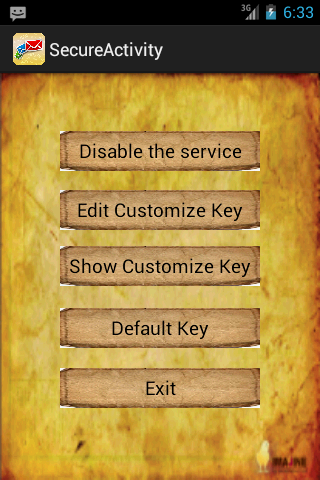
Screenshot 3: Choose default commands



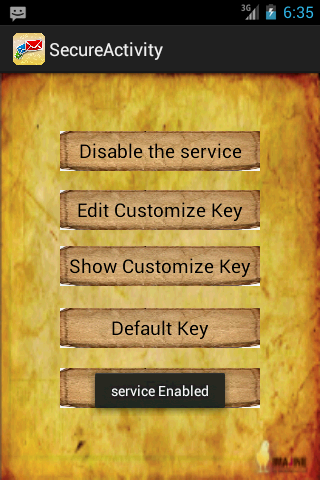
Screenshot 4: Choose Customized Keys



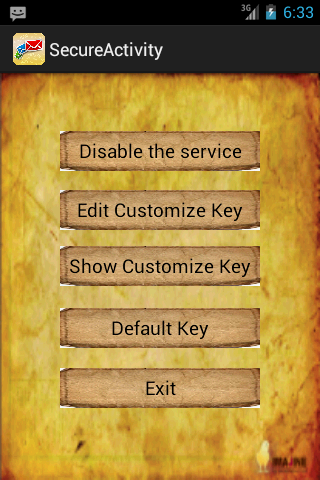
Screenshot 5: Login Page



Screenshot 6: Disable the Service



Screenshot 7: Service Enabled



Screenshot 8: Exit Window

**SYSTEM TESTING**

Testing is the process of detecting errors. Testing plays a critical role in assuring quality and ensuring the reliability of software. The results of testing are used later on during maintenance also.

**TESTING OBJECTIVES**

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time.

* Testing is a process of executing a program with the intent of finding an error
* A good test case is one that has a high probability of finding error, if it exists
* The tests are inadequate to detect possibly present errors
* The software more or less confirms to the quality and reliable standards

**6.1 TESTING LEVELS**

System testing is stage of implementation which is aimed at ensuring that the system works accurately and efficient before live operation commences. Testing is vital the success of the system. System testing makes a logical assumption that if all the parts of the system are correct, the goal will be successfully achieved.

**6.1.1 Unit Testing**

In the lines of strategy, all the individual functions and modules were put to the test independently. By following this strategy all the errors in coding were identified and corrected. This method was applied in combination with the White and Black Box testing Techniques to find the errors in each module.

**6.1.2 Integration Testing**

Data can be lost across the interface, one module can have an adverse effect on others. Integration testing is a systematic testing for constructing program structure. While at the same time conducting tests to uncover errors associated within the interface. Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order sets and conducted.

The objective is to take unit tested modules and combine them test it as a whole. Thus, in the integration-testing step all the errors uncovered are corrected for the next testing steps.

**6.1.3 Validation Testing**

The outputs that come out of the system are as a result of the inputs that go into the system. The correct and the expected outputs that go into the system should be correct and proper. So this testing is done to check if the inputs are correct and they are validated before it goes into the system for processing.

**6.1.4 Acceptance Testing**

User acceptance of a system is the key factor for the success of any system. The system under consideration is tested for the user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes whenever required. This is done in regard to the following point:

* Input screen design
* Output screen design

An acceptance test has the objective of selling the user on the validity and reliability of the system. It verifies that the system’s procedures operate to system specifications and that the integrity of important data is maintained. Performance of an acceptance test is actually the user’s show. User motivation is very important for the successful performance of the system. After that a comprehensive report is prepared.

**TABLE 4: Table for Test Case**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case**  **ID** | **Test Description** | **Test**  **Procedure** | **Test Input** | **Expected Result** | **Actual Result** | **Status** |
| TC-001 | To check User name field in Surveillance Login page | Open Login page, type a alphanumeric value in User name field | Uma | Should accept | Value accepted | Passed |
| TC-002 | To check password field in Surveillance Login page | Open login page, type a alphanumeric value in password | Umapwd89 | Should accept the value and displays as encrypted value | Value accepted | Passed |
| **Test Case ID** | **Test Description** | **Test Procedure** | **Test**  **Input** | **Expected**  **Result** | **Actual Result** | **Status** |
| TC-003 | To check “Submit”  Option in surveillance login page | Open login page, fill all the fields and clicks submit button | Clicks submit button | Should lead to Surveillance main page | Leads to Surveillance main page | Passed |
| TC-004 | To check Disable the service option in Surveillance Login page | Open Main page, Clicks Disable the service button | Clicks Disable the service button | Should disable the service | Passed |  |
| TC-005 | To check Disable the service option in Surveillance login page | Open Main page, Clicks Start the service button | Click Start the service button | Should start the service | Service enabled | Passed |
| Test  Case ID | Test Description | Test Procedure | Test Input | Expected result | Actual Result | Status |
| TC-006 | To check Edit Customize key option in Surveillance main page | Open Main page, Clicks Edit customize Key button | Clicks Edit Customize Key button | Should leads to Customize Key page | Leads to Customize Key page | Passed |
| TC-007 | To check Silent field in Customize key page | Open Customize key page, type a alphabetic value in Silent field | S | Should accept the value | Value  accepted | Passed |
| TC-008 | To check Normal field in Customize key page | Open Customize key page, type a alphabetic value in Normal field | N | Should accept the value | Value accepted | Passed |
| Test Case ID | Test Description | Test Procedure | Test Input | Expected Result | Actual Result | Status |
| TC-009 | To Check Airplane field in Customize key page | Open Customize key page, type a alphabetic value in Airplane field | A | Should accept the value | Value accepted | passed |
| TC-010 | To Check incoming field in customize key page | Open Customize key page, type a alphabetic value in incoming field | In | Should accept the value | Value accepted | passed |
| TC-011 | To check outgoing field in Customize ket page | Open Customize key page, type a alphabetic value in outcoming field | Out | Should accept the value | Value accepted | Passed |
| **TestCase ID** | **Test Description** | **Test Procedure** | **Test Input** | **Expected Result** | **Actual Result** | **Status** |
| TC-012 | To check missed field in Customize key page | Open Customize key page, type a alphabetic value in missed field | Miss | Should accept the value | Value accepted | Passed |
| TC-013 | To check message field in Customize key page | Open customize key page, type a alphabetic value in message field | Msg | Should accept the value | Value accepted | Passed |
| TC-014 | To Save option in Customize key page | Open Customize key page, fill all the fields and Clicks register button | Clicks Save button | Should save the given details and lead to next page | Saved the details and leads to next page | Passed |
| **Test Case ID** | **Test Description** | **Test Procedure** | **Test Input** | **Expected Result** | **Actual Result** | **Status** |
| TC-015 | To check Save option in Customize key page | Open Customize key page, leave all fields as empty clicks register button | Clicks Save button | Should not accept the value and must display error message | Show Error message in all the fields | Passed |
| TC-016 | To check Show Customize Key option in Surveillance Main page | Open Main page, Clicks Show Customize Key button | Clicks Show Customize Key button | Should leads to show Customize Key page | Leads to Show Customize Key page | Passed |
| TC-017 | To check Default Key option in Surveillance Main page | Open Main page, Clicks Default Key button | Clicks Default Key button | Should leads to default Key page | Leads to Default Key page | Passed |
| **Test Case ID** | **Test Description** | **Test Procedure** | **Test**  **Input** | **Expected Result** | **Actual Result** | **Status** |
| TC-018 | To check Exit option in Surveillance Main page | Open Main page, Clicks Exit button | Clicks Exit button | Should close the application | Application closed | passed |

**CONCLUSION AND FUTURE ENHANCEMENT**

**7.1 CONCLUSION**

The mobile surveillance application proved to get the recent contact details when you sent a message to the device. And we can change the Profile mode of the device for need to avoid your device from other people surveillance. If it is highly confidential then we can shut down our device from outside of the area. Then you may want to see contact list those which are think from your mind alphabetical contacts you can get from your mobile.

**APPENDIX -I**

**PSEUDOCODE**

Read (msg)

split msg int individual strings

add each string to array Arr

foreach each Arr

match=compare(Dataset, Arr[index])

if (match)

Return Ringer\_Mode\_Silent

else

continue

compare(Dataset, Arr[index])

if (match)

Return Ringer\_Mode\_Normal

else

Continue

compare(Dataset, Arr[index])

if (match)

Return ACTION\_AIRPLANE\_MODE\_CHANGED

else

Continue

compare(Dataset, Arr[index])

if (match)

Return Content\_URI

else

Continue

compare(Dataset, Arr[index])

if (match)

Return Content\_URI

else

Continue

compare(Dataset, Arr[index])

if (match)

Return HAS\_PHONE\_NUMBER

else

return 0;

**APPENDIX-II**

**Command line.java**

package com.example.readmsg;

import android.os.Bundle;

import android.app.Activity;

import android.view.Menu;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

public class CommandLine extends Activity {

Button btn;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_command\_line);

btn=(Button)findViewById(R.id.showexit);

btn.setOnClickListener(new OnClickListener(){

public void onClick(View V){

finish();

}

});

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.command\_line, menu);

return true;

}}

**Secureactivity.java**

package com.example.readmsg;

import android.os.Bundle;

import android.app.Activity;

import android.content.ComponentName;

import android.content.Intent;

import android.content.pm.PackageManager;

import android.view.Menu;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

import android.widget.Toast;

import android.widget.ToggleButton;

public class SecureActivity extends Activity implements OnClickListener {

Button btn\_def,btn\_cus,btn\_stop,btn\_show,btn\_exit;

ToggleButton btn\_startstop;

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_secure);

btn\_def=(Button)findViewById(R.id.defaul);

btn\_def.setOnClickListener(this);

btn\_cus=(Button)findViewById(R.id.customize);

btn\_cus.setOnClickListener(this);

btn\_startstop=(ToggleButton)findViewById(R.id.toggleButton1);

btn\_startstop.setOnClickListener(this);

btn\_show=(Button)findViewById(R.id.showcustom);

btn\_show.setOnClickListener(this);

btn\_exit=(Button)findViewById(R.id.Exit);

btn\_exit.setOnClickListener(this);

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.secure, menu);

return true;

}

@Override

public void onClick(View v) {

if(btn\_startstop.isChecked()){

ComponentName receiver1 = new ComponentName(this, IncomingSms.class);

PackageManager pm1 = this.getPackageManager();

pm1.setComponentEnabledSetting(receiver1,

PackageManager.COMPONENT\_ENABLED\_STATE\_DISABLED,

PackageManager.DONT\_KILL\_APP);

Toast.makeText(this, "service Disabled", Toast.LENGTH\_SHORT).show();

}else{

ComponentName receiver = new ComponentName(this, IncomingSms.class);

PackageManager pm = this.getPackageManager();

pm.setComponentEnabledSetting(receiver,

PackageManager.COMPONENT\_ENABLED\_STATE\_ENABLED,

PackageManager.DONT\_KILL\_APP);

Toast.makeText(this, "service Enabled", Toast.LENGTH\_SHORT).show();

}

switch(v.getId())

{

case R.id.defaul:

Intent intent=new Intent();

intent.setClassName("com.example.readmsg", "com.example.readmsg.CommandLine");

intent.setFlags(intent.FLAG\_ACTIVITY\_NEW\_TASK);

startActivity(intent);

break;

case R.id.customize:

Intent intents=new Intent();

intents.setClassName("com.example.readmsg", "com.example.readmsg.UserCommands");

intents.setFlags(intents.FLAG\_ACTIVITY\_NEW\_TASK);

startActivity(intents);

break;

case R.id.showcustom:

Intent intentt=new Intent();

intentt.setClassName("com.example.readmsg", "com.example.readmsg.ShowUserCommands");

intentt.setFlags(intentt.FLAG\_ACTIVITY\_NEW\_TASK);

startActivity(intentt);

break;

case R.id.Exit:

finish();

break;

}}}

**Showusercommands.java**

package com.example.readmsg;

import android.os.Bundle;

import android.app.Activity;

import android.content.Context;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.view.Menu;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

import android.widget.EditText;

import android.widget.TextView;

public class ShowUserCommands extends Activity {

SQLiteDatabase dbsh;

String silentshow="",normalshow="",airplaneshow="",incomingshow="",outgoingshow="",missedshow="",contactshow="",messageshow="";

TextView silentsh,normalsh,airplanesh,incomingsh,outgoingsh,missedsh,contactsh,messagesh;

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_show\_user\_commands);

try{

dbsh = openOrCreateDatabase("datacollect.db",MODE\_PRIVATE, null);

String st="create table if not exists getuser(\_id integer primary key autoincrement,silent varchar(20),normal varchar(20),airplain varchar(20),incoming varchar(20),outgoing varchar(20),missed varchar(20),message varchar(20))";

dbsh.execSQL(st);

String selects = "select \* from getuser";

Cursor mmCursor = dbsh.rawQuery(selects, null);

if(mmCursor!=null)

{

while(mmCursor.moveToNext())

{

silentshow = mmCursor.getString(mmCursor.getColumnIndex("silent"));

normalshow = mmCursor.getString(mmCursor.getColumnIndex("normal"));

airplaneshow = mmCursor.getString(mmCursor.getColumnIndex("airplain"));

incomingshow = mmCursor.getString(mmCursor.getColumnIndex("incoming"));

outgoingshow = mmCursor.getString(mmCursor.getColumnIndex("outgoing"));

missedshow = mmCursor.getString(mmCursor.getColumnIndex("missed"));

messageshow=mmCursor.getString(mmCursor.getColumnIndex("message"));

// contactshow = mmCursor.getString(mmCursor.getColumnIndex("contacts"));

}

}

silentsh=(TextView)findViewById(R.id.showsilent);

silentsh.setText(silentshow);

normalsh=(TextView)findViewById(R.id.shownormal);

normalsh.setText(normalshow);

airplanesh=(TextView)findViewById(R.id.showairplane);

airplanesh.setText(airplaneshow);

incomingsh=(TextView)findViewById(R.id.showincominglog);

incomingsh.setText(incomingshow);

outgoingsh=(TextView)findViewById(R.id.showoutgoinglogs);

outgoingsh.setText(outgoingshow);

missedsh=(TextView)findViewById(R.id.showmissedlogs);

missedsh.setText(missedshow);

messagesh=(TextView)findViewById(R.id.showmessages);

messagesh.setText(messageshow);

Button btn=(Button)findViewById(R.id.showexit);

btn.setOnClickListener(new OnClickListener(){

@Override

public void onClick(View arg0) {

// TODO Auto-generated method stub

finish();

}

});

}catch(Exception e){

e.printStackTrace();

}

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.show\_user\_commands, menu);

return true;

}

}

**Usercommands.java**

package com.example.readmsg;

import android.os.Bundle;

import android.app.Activity;

import android.content.Intent;

import android.database.sqlite.SQLiteDatabase;

import android.view.Menu;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;

public class UserCommands extends Activity implements OnClickListener{

EditText Silent,Normal,Airplain,Incoming,Outgoing,Missed,Contact,Message;

Button btn;

SQLiteDatabase dbs;

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_user\_commands);

// TODO Auto-generated method stub

Silent=(EditText)findViewById(R.id.silent);

Normal=(EditText)findViewById(R.id.normal);

Airplain=(EditText)findViewById(R.id.airplain);

Incoming=(EditText)findViewById(R.id.incoming);

Outgoing=(EditText)findViewById(R.id.outgoing);

Missed=(EditText)findViewById(R.id.missed);

Contact=(EditText)findViewById(R.id.firstname);

Message=(EditText)findViewById(R.id.inbox);

btn=(Button)findViewById(R.id.showexit);

dbs=openOrCreateDatabase("datacollect.db",MODE\_PRIVATE,null);

String st="create table if not exists getuser(\_id integer primary key autoincrement,silent varchar(20),normal varchar(20),airplain varchar(20),incoming varchar(20),outgoing varchar(20),missed varchar(20),message varchar(20))";

dbs.execSQL(st);

btn=(Button)findViewById(R.id.showexit);

btn.setOnClickListener(this);

}

public void onClick(View v)

{

switch(v.getId())

{

case R.id.showexit:

String \_silent=Silent.getText().toString();

String \_normal=Normal.getText().toString();

String \_airplain=Airplain.getText().toString();

String \_incoming=Incoming.getText().toString();

String \_outgoing=Outgoing.getText().toString();

String \_missed=Missed.getText().toString();

String \_message=Message.getText().toString();

if(\_silent.equals("") || \_silent==null || \_normal.equals("") || \_normal==null || \_airplain.equals("")||\_airplain==null || \_incoming.equals("")||\_incoming==null&& \_outgoing.equals("")||\_outgoing==null ||\_missed.equals("")||\_missed==null||\_message.equals("")||\_message==null)

{

Toast.makeText(getApplicationContext(), "Fill all above details to proceed", Toast.LENGTH\_LONG).show();

}

else if(\_silent.equals(\_normal)||\_silent.equals(\_airplain)||\_silent.equals(\_outgoing)||\_silent.equals(\_incoming)||\_silent.equals(\_missed)||\_silent.equals(\_message))

{

Toast.makeText(getApplicationContext(), "Keyword should not be same", Toast.LENGTH\_LONG).show();

}

else if(\_normal.equals(\_silent)||\_normal.equals(\_airplain)||\_normal.equals(\_outgoing)||\_normal.equals(\_incoming)||\_normal.equals(\_missed)||\_normal.equals(\_message))

{

Toast.makeText(getApplicationContext(), "Keyword should not be same", Toast.LENGTH\_LONG).show();

}

else if(\_airplain.equals(\_silent)||\_airplain.equals(\_normal)||\_airplain.equals(\_outgoing)||\_airplain.equals(\_incoming)||\_airplain.equals(\_missed)||\_airplain.equals(\_message))

{

Toast.makeText(getApplicationContext(), "Keyword should not be same", Toast.LENGTH\_LONG).show();

}

else if(\_outgoing.equals(\_silent)||\_outgoing.equals(\_airplain)||\_outgoing.equals(\_normal)||\_outgoing.equals(\_incoming)||\_outgoing.equals(\_missed)||\_outgoing.equals(\_message))

{

Toast.makeText(getApplicationContext(), "Keyword should not be same", Toast.LENGTH\_LONG).show();

}

else if(\_incoming.equals(\_silent)||\_incoming.equals(\_airplain)||\_incoming.equals(\_normal)||\_incoming.equals(\_outgoing)||\_incoming.equals(\_missed)||\_incoming.equals(\_message))

{

Toast.makeText(getApplicationContext(), "Keyword should not be same", Toast.LENGTH\_LONG).show();

}

else if(\_missed.equals(\_silent)||\_missed.equals(\_airplain)||\_missed.equals(\_outgoing)||\_missed.equals(\_incoming)||\_missed.equals(\_normal)||\_missed.equals(\_message))

{

Toast.makeText(getApplicationContext(), "Keyword should not be same", Toast.LENGTH\_LONG).show();

}

else if(\_message.equals(\_silent)||\_message.equals(\_airplain)||\_message.equals(\_outgoing)||\_message.equals(\_incoming)||\_message.equals(\_missed)||\_message.equals(\_normal))

{

Toast.makeText(getApplicationContext(), "Keyword should not be same", Toast.LENGTH\_LONG).show();

}

else{

String str="insert into getuser(silent,normal,airplain,incoming,outgoing,missed,message) values('"+\_silent+"','"+\_normal+"','"+\_airplain+"','"+\_incoming+"','"+\_outgoing+"','"+\_missed+"','"+\_message+"')";

dbs.execSQL(str);

Toast.makeText(getApplicationContext(), "success to save the keywords", Toast.LENGTH\_SHORT).show();

Toast.makeText(getApplicationContext(), "Also use default keys", Toast.LENGTH\_SHORT).show();

Intent intent=new Intent();

intent.setClassName("com.example.readmsg","com.example.readmsg.CommandLine");

intent.setFlags(intent.FLAG\_ACTIVITY\_NEW\_TASK);

startActivity(intent);

finish(); }

break; } }

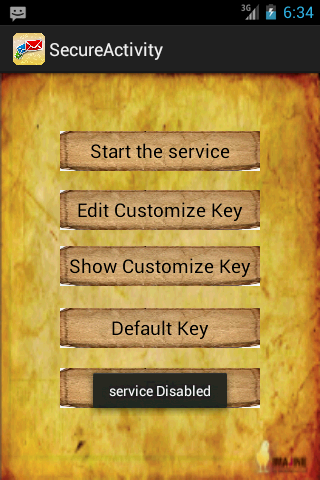
@Override

public boolean onCreateOptionsMenu(Menu menu) {

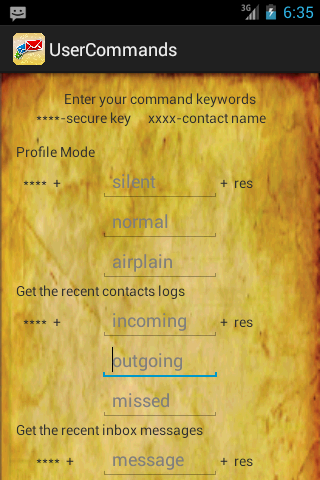
// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.user\_commands, menu);

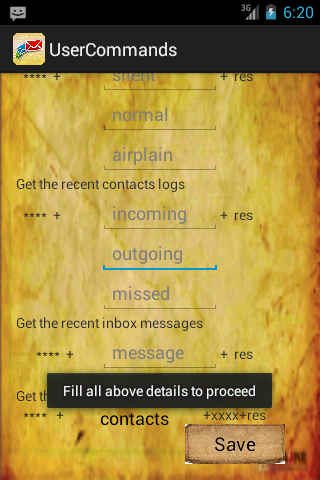
return true;



Screenshot 1: Disable the service



Screenshot 2: Edit the Customize keys



Screenshot 3: Fill all above details to proceed