PROJECT REPORT
ON

“Call Proxy”

SPONSORED BY

“Persistent Systems Limited”

BY

Deshpande Tejashree Vilas

Kothari Sayali Sunil

Lunkad Shruti Anil

Shah Nitee Arvind

CUMMINS COLLEGE OF ENGINEERING FOR WOMEN

PUNE – 411052

* 2011 – 2012 *

PROJECT REPORT

ON

“Call Proxy”

SPONSORED BY

“Persistent Systems Limited”

BY

Deshpande Tejashree Vilas

Kothari Sayali Sunil

Lunkad Shruti Anil

Shah Nitee Arvind

CUMMINS COLLEGE OF ENGINEERING FOR WOMEN

PUNE – 411052

* 2011 – 2012 *

CUMMINS COLLEGE OF ENGINEERING FOR WOMEN

KARVENAGAR, PUNE 411052.

CERTIFICATE

This is to certify that the following students

Ms. Deshpande Tejashree Vilas

Ms. Kothari Sayali Sunil

Ms. Lunkad Shruti Anil

Ms. Shah Nitee Arvind

Have completed the Project entitled

"Call Proxy"

Satisfactory for the partial fulfillment of the requirements for the Bachelor's Degree in Information Technology of Pune University during Academic Year 2010 – 2011.

Name:	Name:
Mr. Lovenish Parvani	Ms. Harsha Khedkar
Prof. (Mrs.) Anagha Kulkarni	Dr. Mrs. Madhuri Khambate
Head of Department	Principal

Cummins College of Engineering For Women
Department of Information Technology
Pune – 411052.

CERTIFICATE

This is to certify that the Dissertation entitled "**Call Proxy**", submitted by

Ms. Tejashree Vilas Deshpande
Ms. Sayali Sunil Kothari
Ms. Shruti Anil Lunkad
Ms. Nitee Arvind Shah

is a record of bonafide work carried out by above listed students, in the partial fulfillment of the requirement for the award of Degree of Bachelor of Engineering(Information Technology) at Cummins College of Engineering For Women, Pune under the University of Pune. This work is done during year 2011-12, under our guidance.

Date : ……………….

Project Guide [Prof. Ms. Harsha Khedkar]

--- --
[Prof. Anagha Kulkarni] [Dr. Madhuri Khambete]
HOD, Information Technology Principal, Cummins COEW

Examination

Examiner 1:………..……………………………………………………………………

Examiner 2:…..………………………………………………………………………

ACKNOWLEDGEMENT

We would like to thank our Internal Guide**, Ms. Harsha Khedkar, Cummins College of Engineering for Women, Pune** for her able guidance and useful suggestions, which helped us in the completion of this project work.

We take immense pleasure in thanking **our H.O.D., Mrs. Anagha Kulkarni** for having permitted us to carry out this project work.

We would like to express our deepest gratitude to **Mr. Lovenish Parwani**, Team Lead, Enterprise and Business Solutions, Persistent Systems Ltd. for his effective guidance in developing this mobile application. We would like to thank **Mr. Surendra Sharma** for his timely guidance.

We would also like to thank **Prof. Harshad Wadkar** for his guidance and motivation.

We would like to take this opportunity to sincerely thank **Mr. Lavjeet Singh**, for all the help he has provided throughout the completion of this project.

Finally, yet importantly, we would like to express our heartfelt thanks to our friends/classmates for their help and wishes for the successful commencement of this project.

Deshpande Tejashree Vilas
Kothari Sayali Sunil
Lunkad Shruti Anil
Shah Nitee Arvind

ABSTRACT

There is a rise in using Android OS by the key players of the mobile market. Hence we are developing the mobile application on this platform. This app will help the users manage their calls effectively, by blocking calls and SMS from Blacklisted numbers. Call Proxy will also block calls based on time based profiles, which the user can configure according to their needs. It also creates fake space i.e. a virtual profile on the cell phone which hides the vital details from the unauthenticated user. Our aim is to develop an aesthetically pleasing and user friendly system. We are also striving to make this application power effective. The app will support the backup application data.

According to the Android market, the existing applications support call blocking and managing of privacy. But, all these features are available only at premium costs. So we have decided to add dynamic features to the existing application and provide it free of cost to all the customers.

The software is an advanced call manager with the following objectives:

- To block black listed calls
- To accept calls only from white listed numbers.
- To create a fake profile for the phone.
- To erase selective log entries permanently.
- To create a back-up of phone memory as well as installed apps.

1. Introduction

1.1. Android OS

Android is a Linux-based operating system for mobile devices such as smartphones and tablet computers. It is developed by the Open Handset Alliance led by Google.

Features :

- Application framework enabling reuse and replacement of components
- Dalvik virtual machine optimized for mobile devices
- Integrated browser based on the open source WebKit engine
- Optimized graphics powered by a custom 2D graphics library; 3D graphics based on the OpenGL ES 1.0 specification (hardware acceleration optional)
- SQLite for structured data storage
- Media support for common audio, video, and still image formats (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
- GSM Telephony
- Bluetooth, EDGE, 3G, and WiFi
- Camera, GPS, compass, and accelerometer
- Rich development environment including a device emulator, tools for debugging, memory and performance profiling, and a plugin for the Eclipse IDE

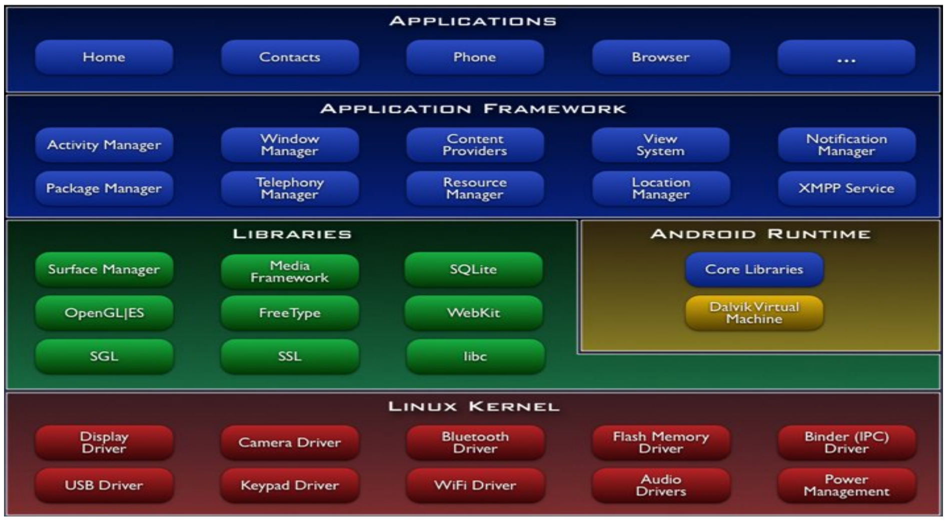


Figure 1: Android Architecture

The above figure shows the android architecture. By providing an open development platform, Android offers developers the ability to build extremely rich and innovative applications. Developers are free to take advantage of the device hardware, access location information, run background services, set alarms, add notifications to the status bar, and much, much more.

Developers have full access to the same framework APIs used by the core applications. The application architecture is designed to simplify the reuse of components; any application can publish its capabilities and any other application may then make use of those capabilities (subject to security constraints enforced by the framework). This same mechanism allows components to be replaced by the user.

1.2. Introduction to Call Proxy

This is an Advanced call manager application for managing calls effectively through the busy schedule. This app is helpful for the users to protect their confidential SMS and call logs from unwanted people spying through the phone.

Call Proxy is app for the android users for efficiently managing the calls and SMS. This app lets you create white lists and blacklist and on selection, lets the user accept calls from white list and reject calls and SMS from blacklist. Along with this, it lets the user create Time based Profiles. The Time based profile creates white list and automatically activates it based on the time assigned to that profile.

Private Space lets the user to hide confidential SMS. The SMS and Call logs of Private contacts are stored in a different location in the Private Space that is password protected.

The app also allows the user to Backup the application data and restore it to other device or the same device. The backup is taken on the SD card of the phone.

1.3. Need of Call Proxy

Call proxy helps the users to effectively manage the calls and SMS. The currently available applications are available at a premium cost. We propose on providing these features free of cost.

This app allows the user to store as many contacts in the black list/white list as they want, in contrast to the previously existing applications that allow only a limited contacts in the database. The app uses battery power effectively.

1.4. Scope Definition

The software is an advanced call manager with the following objectives:

- To block black listed calls
- To accept calls only from white listed numbers.
- To erase selective log entries permanently.
- To erase conversations with a particular user.
- To create a back-up of contacts.
- Backup the application data.
- Restore the backed up data.

2. Literature Survey

The first stage in designing a solution to any problem is to determine the requirements of the new system.

2.1. Requirement gathering

Various technologies were used for gathering information regarding the system.

Brainstorming technique:

Here, we had an informal meeting with the expertise and tried to understand the current needs of the android smart phone users.

2.2. Requirement analysis

Requirement analysis started in parallel with requirement gathering and involved refining and modeling the requirements to identify inconsistencies, errors etc. The key objective of requirement analysis is to describe the requirements in terms of relationships, provide a basis for validations for the software after it is build. Various existing applications were reviewed, in order to get a look and feel of the basic user interaction features that we needed to implement in our application.

2.3. Literature Survey

There is a rise in using Android OS by the key players of the mobile market. Hence we have developed the mobile application on this platform. According to the Andriod market, the existing applications support call blocking and managing of privacy. But, all these features are available only at premium costs. So we have added dynamic features to the existing application and are providing it free of cost to all the customers.

The applications whice we reviewed are :

- NetQin Mobile Manger :

  The feature of private space is available only in the premium version. Application is insecured, is not protected by password.
- DND Call Blocker :

  Blocks only black listed calls and messages. No feature of white list.
- Call Manager :

Limits the number of contacts that can be added to the black list/white list. Only premium version supports unlimited entries.

We have gathered basic information regarding the domain from the company website. We have also visited various leading android marketplace websites to look out for existing applications.

3. Work Undertaken

3.1. Problem Definition

This product is a self-contained product, referenced from an existing application. This product helps the mobile users to manage and block unwanted calls and messages effectively, as well as protect the handset by using ‘Private Space’. The application helps the users to manage their phone automatically by creating ‘Time based Profiles’.

This application helps the users manage their calls effectively, by blocking calls and SMS from Blacklisted numbers. Call Proxy also blocks calls based on time based profiles, which the user can configure according to their needs. It also creates a fake space i.e. a virtual profile on the cell phone which hides the vital details from the unauthenticated user. We have looked forward to develop an aesthetically pleasing and user friendly system. We have also made this application power effective. The application supports the backup and restore of application data.

According to the Android market, the existing applications support call blocking and managing of privacy. But, all these features are available only at premium costs. So we have added dynamic features to the existing application and provide it free of cost to all the customers.

3.2. Features

The major functions performed by this application are:

3.2.1. Create Black list :

Create Black lists by adding numbers from Contacts or call logs or User input. There is no limit on the number of contacts that can be added to the White list.

<u>Steps In Creating Black List :</u>

- Log in into the application.
- Select Manage List.

- Select Manage Blacklist.
- Press the option button to add contacts.
- Add contacts either from contacts, call logs or input number.
- To delete a number, long press on it and select delete.

3.2.2. Block calls and messages

The application blocks all the calls from the Black listed number. The log for the blacklisted call will maintained separately. The SMS received from the blacklisted number is stored in the database for the blacklisted SMS.

To enable black listing :

- Enable Black list from block settings

3.2.3. Create White list

Create White lists by adding numbers from Contacts or call logs or User input. There is no limit on the number of contacts that can be added to the White list.

Steps In Creating White List :

- Log in into the application.
- Select Manage List.
- Select Manage White list.
- Press the option button to add contacts.
- Add contacts either from contacts, call logs or input number.
- To delete a number, long press on it and select delete.

3.2.4. Receive calls and messages only from white listed numbers:

When the white list is active, the user will receive only the calls from the white-listed numbers. This will help the user avoid the distractions from unwanted callers.

To enable white listing :

- Enable White List from block settings

3.2.5. Set time based profiles

The users can create various profiles that can help in effective call management. These profiles are based on time; and activated and deactivated automatically. The user will have to select numbers in the white list for each profile, and this white list will be active during the time assigned to the profile. The user is also given an option to select the days on which the profile should be active.

<u>Steps In Creating White List</u> :

- Log in into the application.
- Select Time Based Profiles.
- Click on the image button ‘ ‘ to create a new profile.
- Add profile name, set start and end time, insert contacts, select days.
- Save the profile.
- To view the profile details click on the ‘ ‘ button and select the required profile.
- To delete a profile, long press on it and select delete.

3.2.6. Privacy Eraser

This allows the users to erase call logs and/or SMS conversations from a particular number.

<u>Steps In Deleting Logs</u> :

- Log in into the application.
- Select Privacy Eraser.
- Long press on the required number.
- Select the required option to delete.

3.2.7. Private Space

The private space stores the call logs and SMS from the selected private number(s) in a different folder. This helps the user to hide confidential messages.

<u>Steps In Adding Contacts to Pricate Space</u> :

- Log in into the application.

- Select Private Space.
- Select ‘Add private contacts’.
- Add profile name, set start and end time, insert contacts, select days.
- Press the option button to add contacts.
- Add contacts either from contacts, call logs or input number.
- To delete a number, long press on it and select delete.
- To view private messages, click on ‘View Private Messages’.
- To view private call logs, click on ‘View Private Call Logs’.
- To delete particular log, long press on it and select delete.
- To reply to a particular message, select the message, click on ‘Reply’, type the message and send.

3.2.8. Back-up and Recovery

The user can Back-up the application data. The backup is taken on the SD card and can be restored on another device.

Steps In Taking Backup :

- Log in into the application.
- Click on Backup/Restore.
- Click on Backup Data.

Steps In Restoring Data :

- Log in into the application.
- Click on Backup/Restore.
- Click on Restore Data.

3.3. Goals

3.3.1. Effective Call Management

This application helps the people driven by schedule to attend important calls and ignore unwanted calls. The user can create time based profiles to manage the people who can contact him in that time span, for the required days.

3.3.2. Optimized Battery usage

Call Proxy runs in the background, hence, will use limited battery power.

The ‘Broadcast Receiver’ that listens to incoming call and SMS is invoked only for 10secs when there is an incoming call or SMS. It performs its task within those 10 seconds and exits.

For time based profiles, there is a service running in the background, which only monitors the current time to start or stop profiles created by the user. Once a profile is activated, the Broadcast Receiver will run only on an incoming call or message for 10 seconds. A service running in background consumes very less battery as compared to an application running in background.

3.3.3. Efficient memory management

The memory is managed effectively by creating SQLite databases to store Black list / White list contacts. SQLite databases are light weight databases, used especially for the Android OS.

3.3.4. Ease of Installation

This application will be uploaded on the android market so that the android users can download and install the app with ease.

3.3.5. User Friendly

We have attempted to make the app user friendly and customizable.

3.4. Objectives :

There is a rise in using Android OS by the key players of the mobile market. Hence we are developing the mobile application on this platform. This app will help the users manage their calls effectively, by blocking calls and SMS from Blacklisted numbers. Call Proxy will also block calls based on time based profiles, that the user can configure according their needs. It also creates fake space i.e. a virtual profile on the cell phone which hides the vital details from the unauthenticated user. Our aim is to develop an aesthetically pleasing and user friendly system.

We are also striving to make this application power effective. The app will support the backup of contacts, calendar entries and messages.

3.5. Constraints

3.5.1. Security considerations

Android system requires all installed applications be digitally signed with a certificate whose private key is held by the application’s developer. The Android system uses the certificate as a means of identifying the author of an application and establishing trust relationships between applications.

Our application will be deployed on the Android Market with an authorized private key. It also provides Login feature to authenticate users.

3.5.2. Java programming standards

- Structure and Documentation
- Naming Conventions
- Recommendations (Rules of Thumb)

4. Project Design

4.1. System Requirement Specification

4.1.1. Hardware Interfaces

The physical characteristics of the application consist of various mobile devices that run the Android OS. Mobile devices contain different hardware specifications and the application requires minimal hardware requirements to operate fully.

4.1.2. Software Interfaces

- Android OS
- Eclipse
- SQLite database - Data is shared using SQLite among software component

4.2. System Architecture

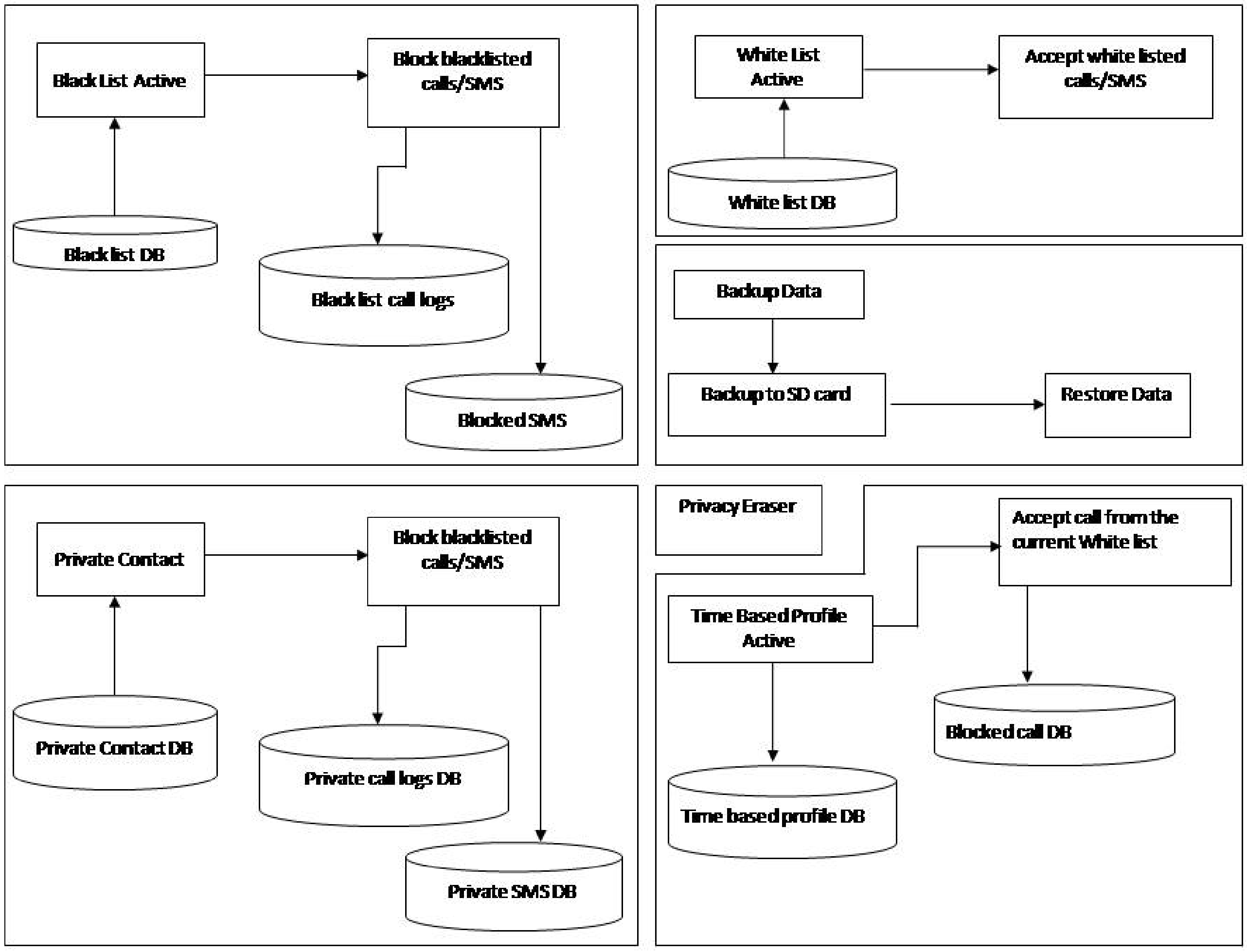


Figure 2: System Architecture

4.3. UML Diagrams

4.3.1. Use Case Diagrams

4.3.1.1. Managing Black list

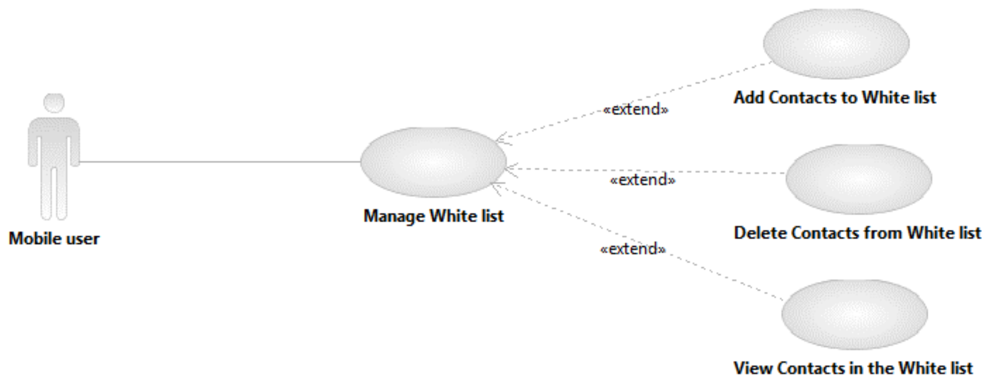

Add Contacts to White list
«extend»
«extend»
Mobile user
Manage White list
Delete Contacts from White list
«extend»
View Contacts in the White list

Figure 3: Use-case for Managing Black list

4.3.1.2. Managing White list

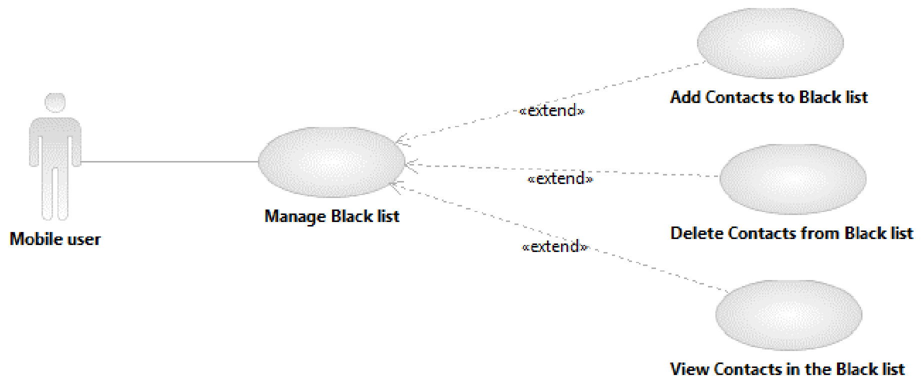

Add Contacts to Black list
«extend»
«extend»
Mobile user
Manage Black list
Delete Contacts from Black list
«extend»
View Contacts in the Black list

Figure 4: Use-case for Managing White list

4.3.1.3. Blocking incoming communication from black listed contacts

Description and Priority

To auto reject calls and messages received from numbers that the user adds in the black list.

Stimulus/Response Sequences

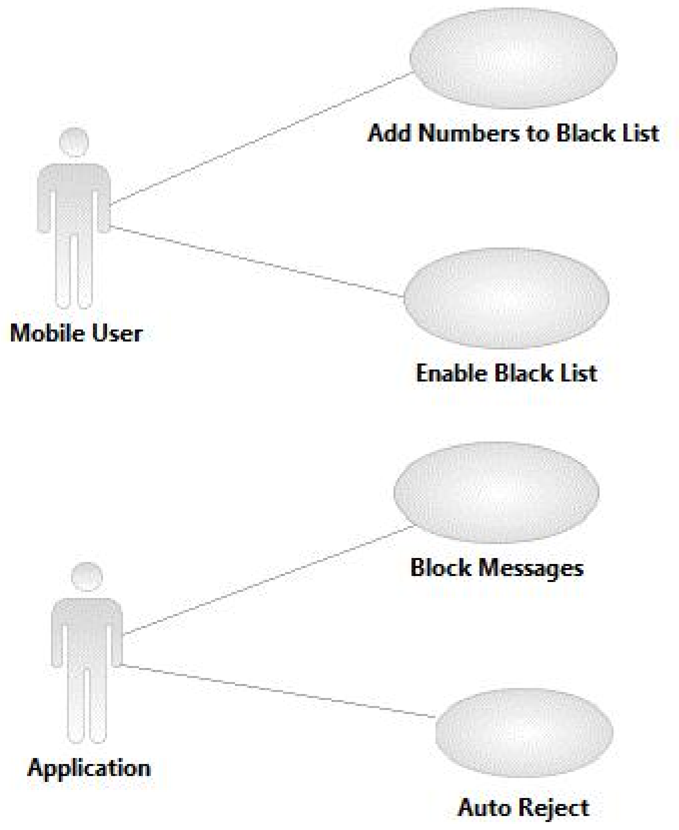


Figure 5: Use-case for Managing Black list

i. The user adds the required numbers in the black list.
ii. The user enables black listing

iii. System filters every incoming message and call from these numbers by auto rejecting calls and moving these messages to a blocked messages folder within the application.

Functional Requirements

i. For the system to auto reject incoming communication, the user must have added contacts to the black list and enabled black listing.

ii. If the list of black listed calls includes the maximum count, the system will delete the oldest logs to make way for the new logs.

REQ-1: Add contacts to the black list
REQ-2: Enable black list

4.3.1.4. Accept incoming communication only from White Listed Numbers

Description and Priority

To accept calls and messages received only from numbers that the user adds in the white list.

Stimulus/Response Sequences

i. The user adds the required numbers in the white list.
ii. The user enables white listing
iii. System accepts only incoming messages and calls from these numbers.

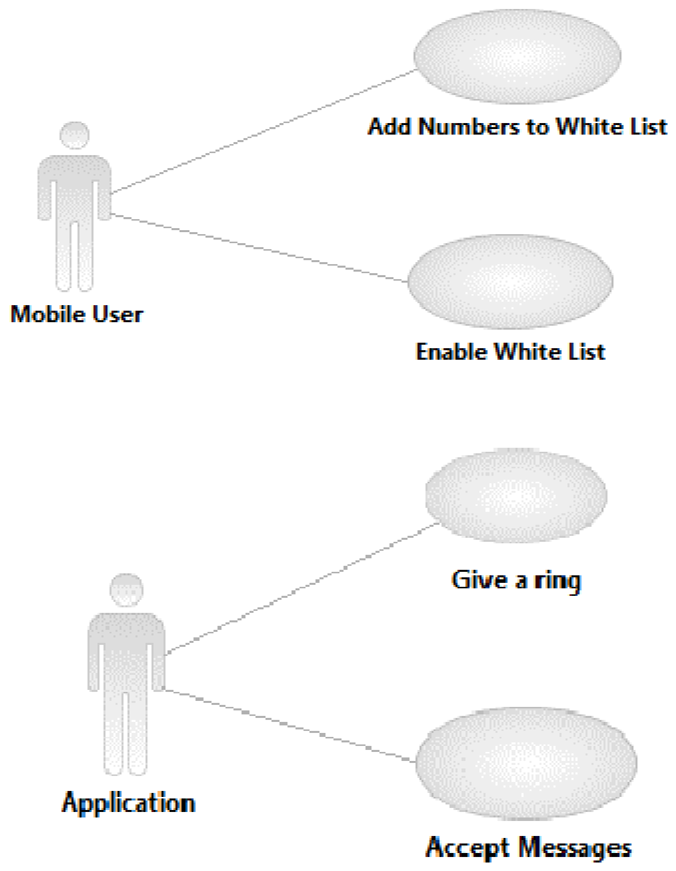


Figure 6: Use-case for Managing White list

Functional Requirements

i. For the system to accept incoming communication, the user must have added contacts to the white list and enabled white listing.

REQ-1: Add contacts to the white list.
REQ-2: Enable white list.

4.3.1.5. Block spam messages

Description and Priority

To block incoming messages from the list of spam numbers.

Stimulus/Response Sequences

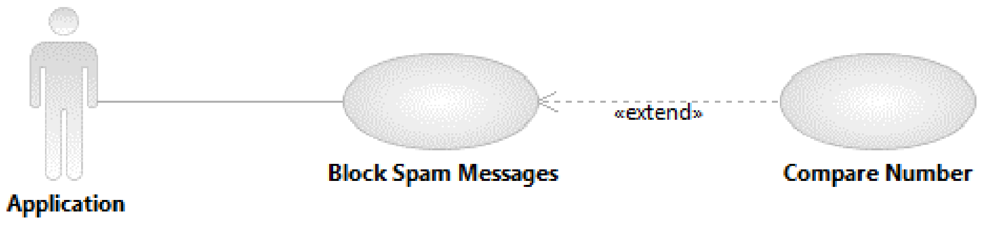


Figure 7:Use-case for Managing Spam Messages

i) For every incoming message the system first compares it with the list of spam numbers, if matched the message is moved to spam folder.

Functional Requirements

The messages will be stored in the spam folder for maximum 20 days, following their arrival and then will be deleted automatically.

4.3.1.6. Privacy Eraser

Description and Priority

The system allows the user to delete the logs from a particular user permanently.

Stimulus/Response Sequences

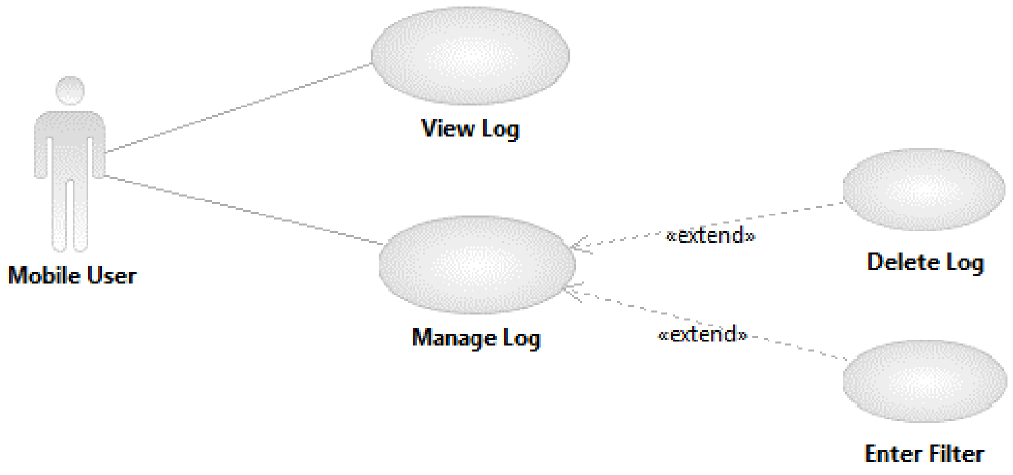


Figure 8:Use-case for Privacy Eraser

i. The user enters the name of the contact or the number.

ii. The system deletes all the logs(messages and calls) from these numbers permanently from the log file.

Functional Requirements

For the system to delete the logs the user must enter the name or number.

REQ-1: Enter the name or number to erase logs associated with it.

4.3.1.7. Backup

Description and Priority

The system allows the user to take back up of contacts on the memory card which can be later restored.

Stimulus/Response Sequences

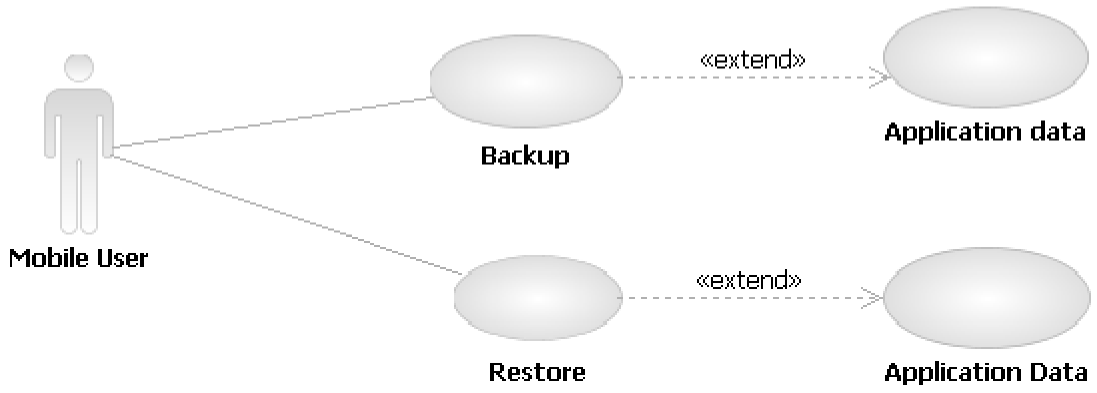


Figure 9: Use-case for Back-up

1. The user selects back up.
2. The system creates a back up file on the memory card.
3. The user selects back up.
4. The system creates a back up file on the memory card.

4.3.2. Activity Diagram

4.3.2.1. White/Black List Contacts

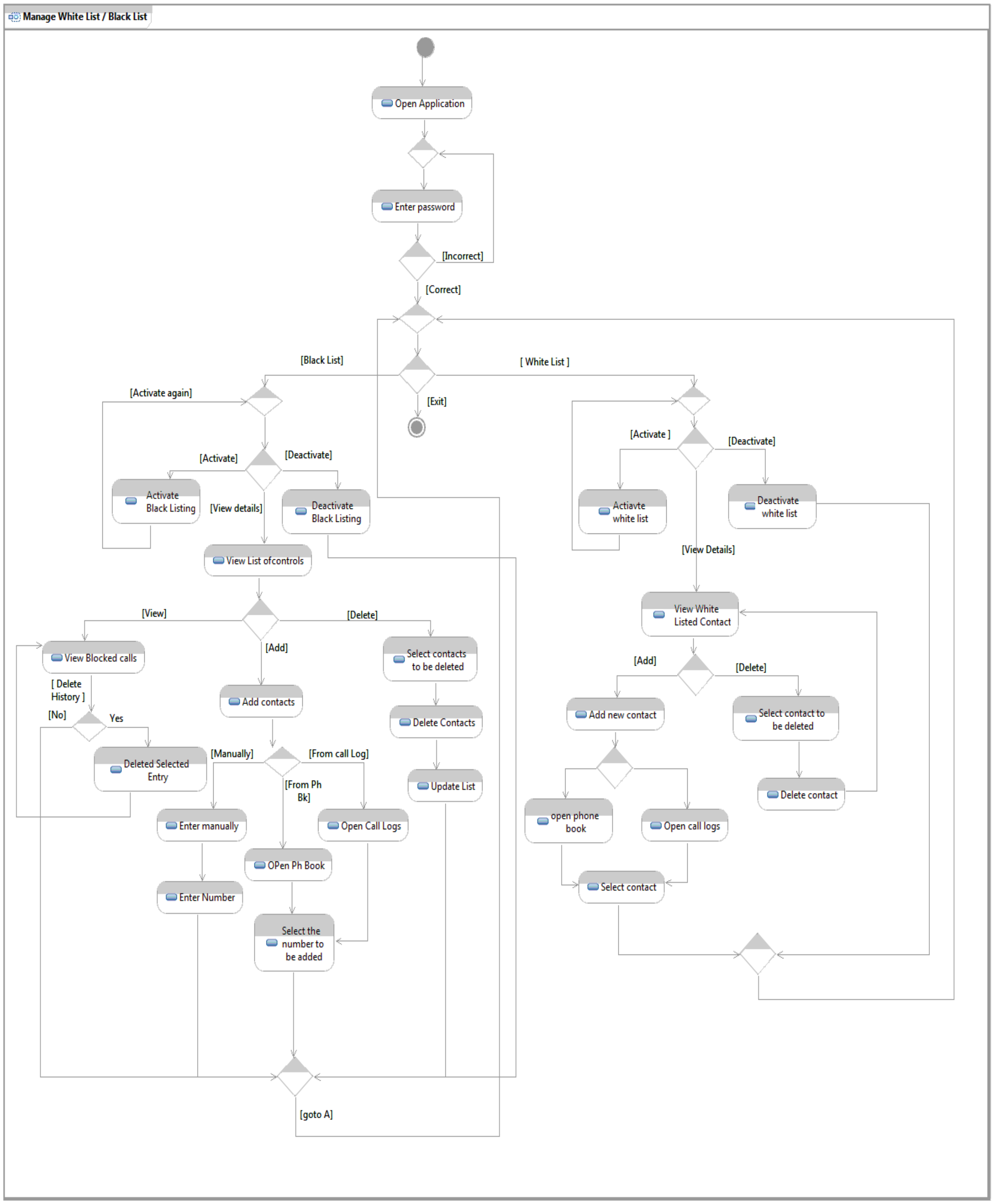


Figure 10 :Activity Diagram: Manage White/Black List Contacts

4.3.2.2. View Blocked Calls/SMS

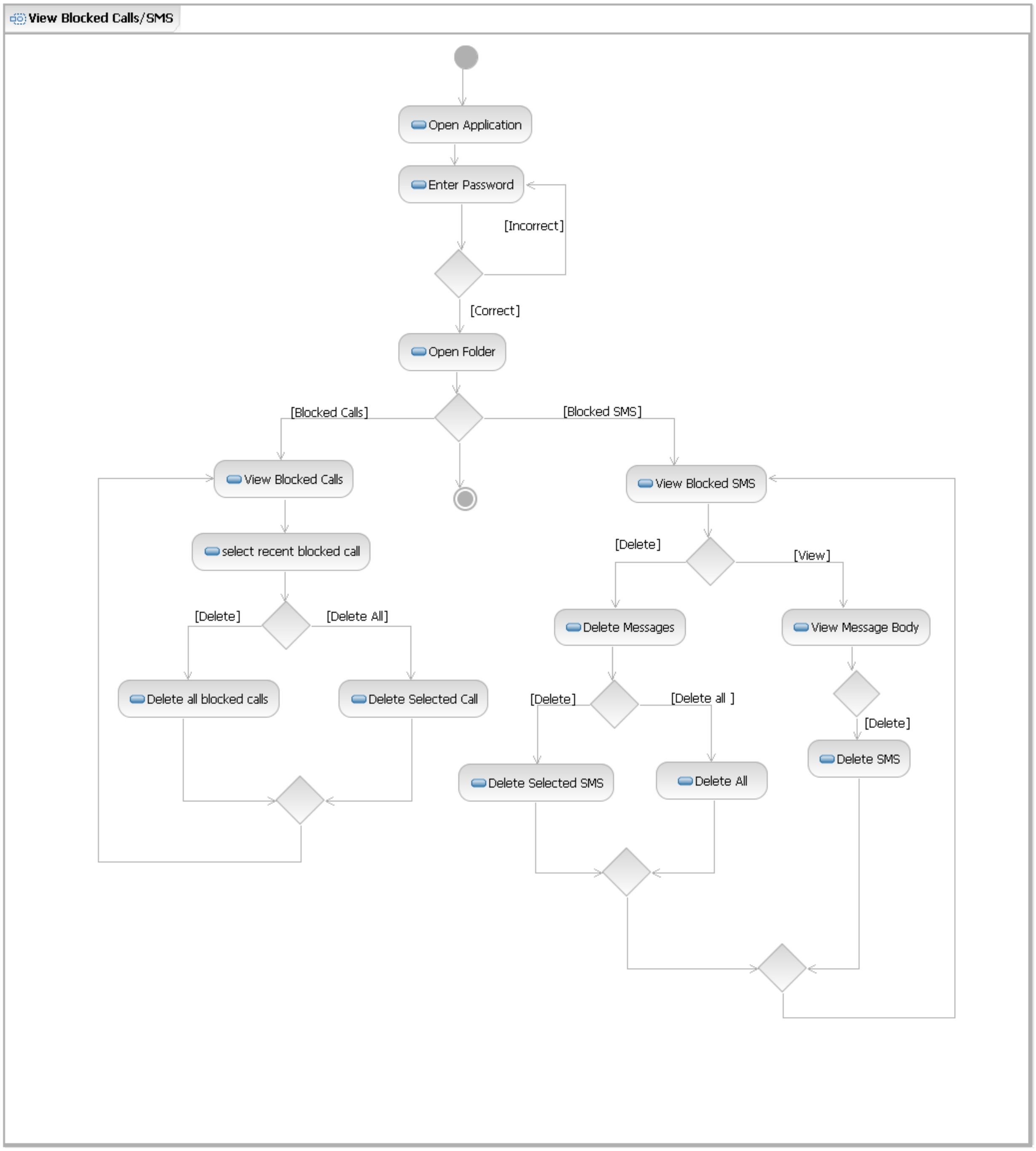


Figure 11: Activity Diagram: Manage Spam Message List

4.3.2.3. Privacy Eraser

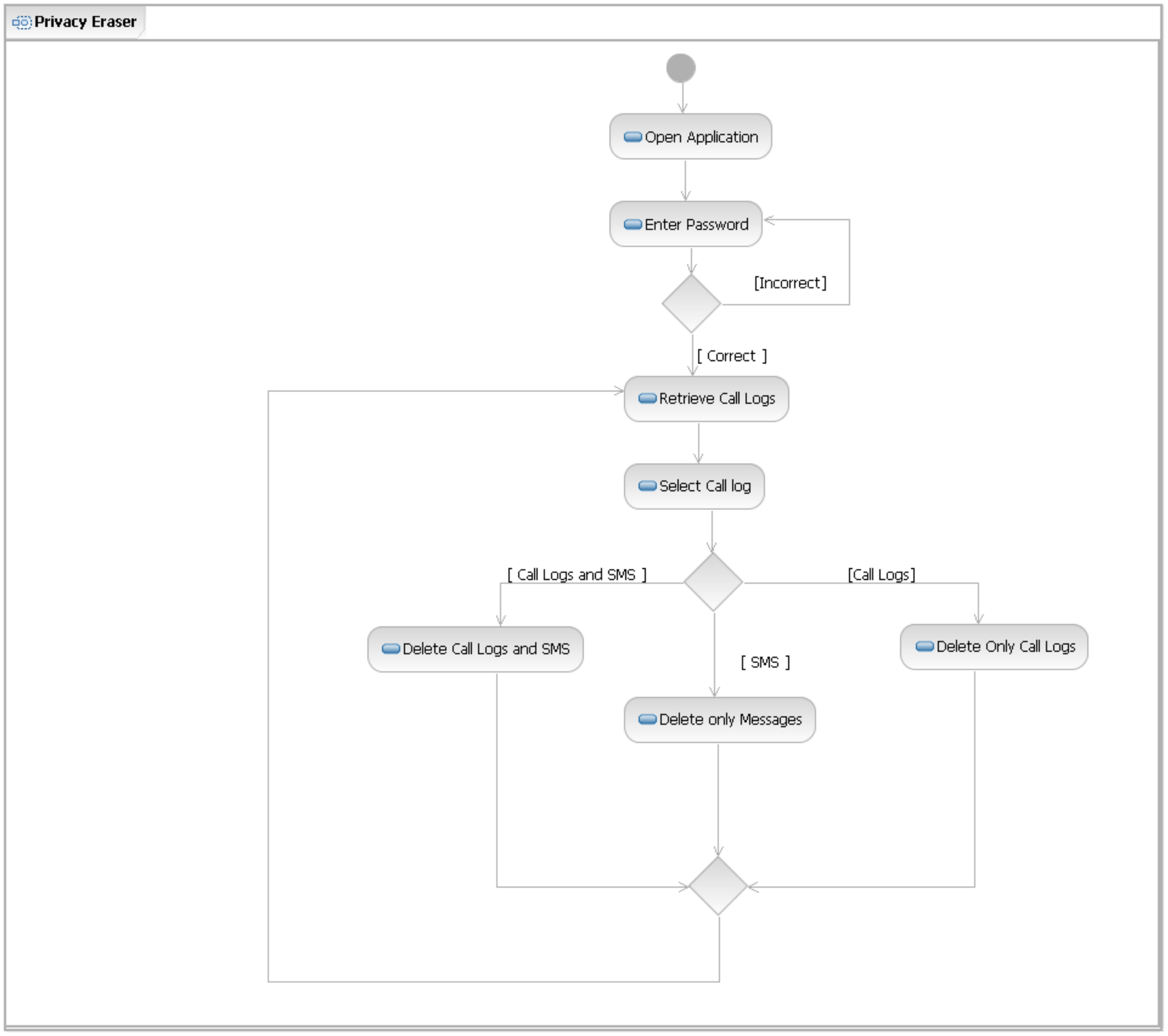


Figure 12 : Activity Diagram Privacy Eraser

4.3.3. Flowchart

4.3.3.1. System Flowchart

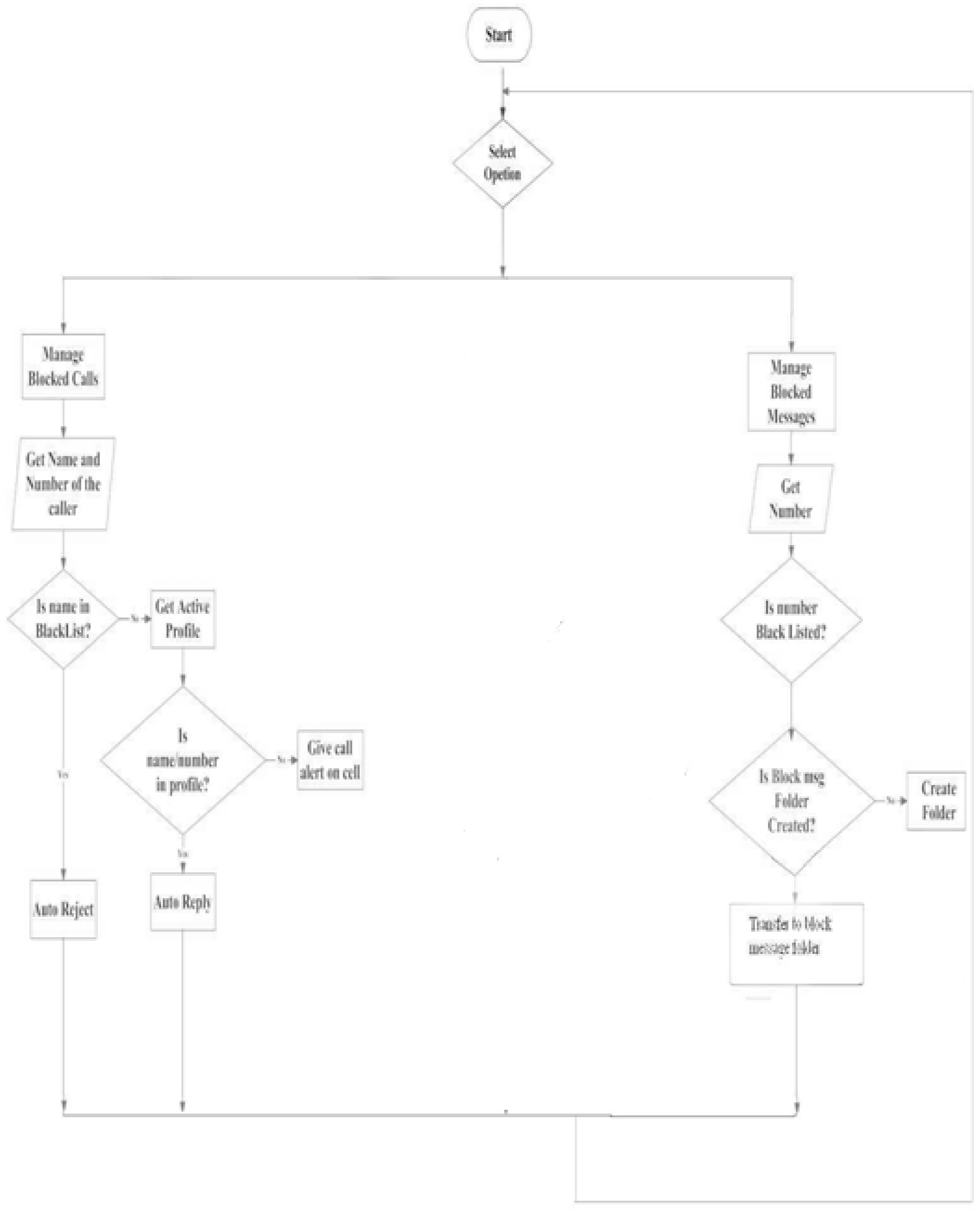


Figure 13: System Flowchart

4.3.3.2. Configuration Flowchart

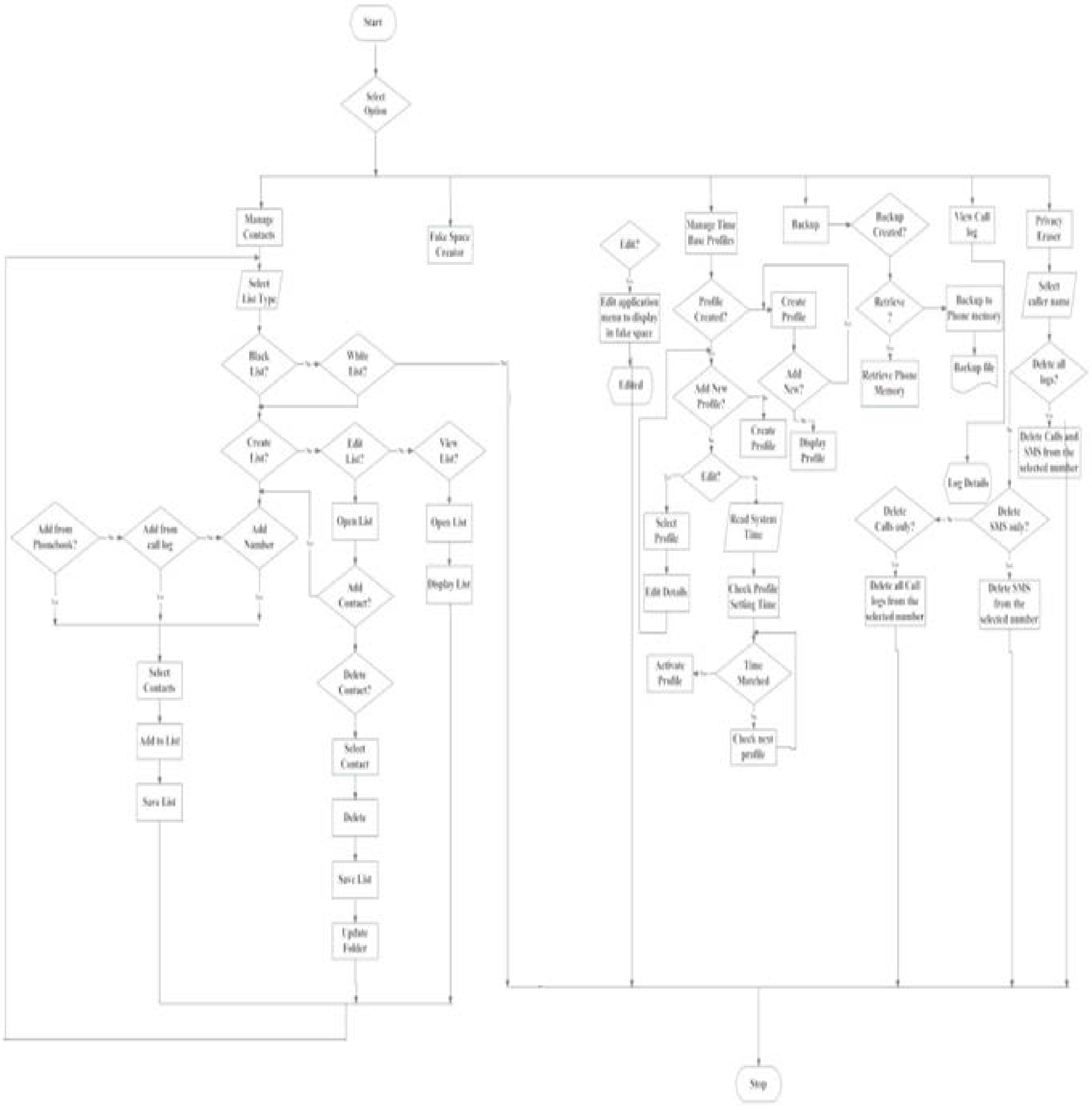


Figure 14: Configuration Flowchart

4.3.4. Class Diagram

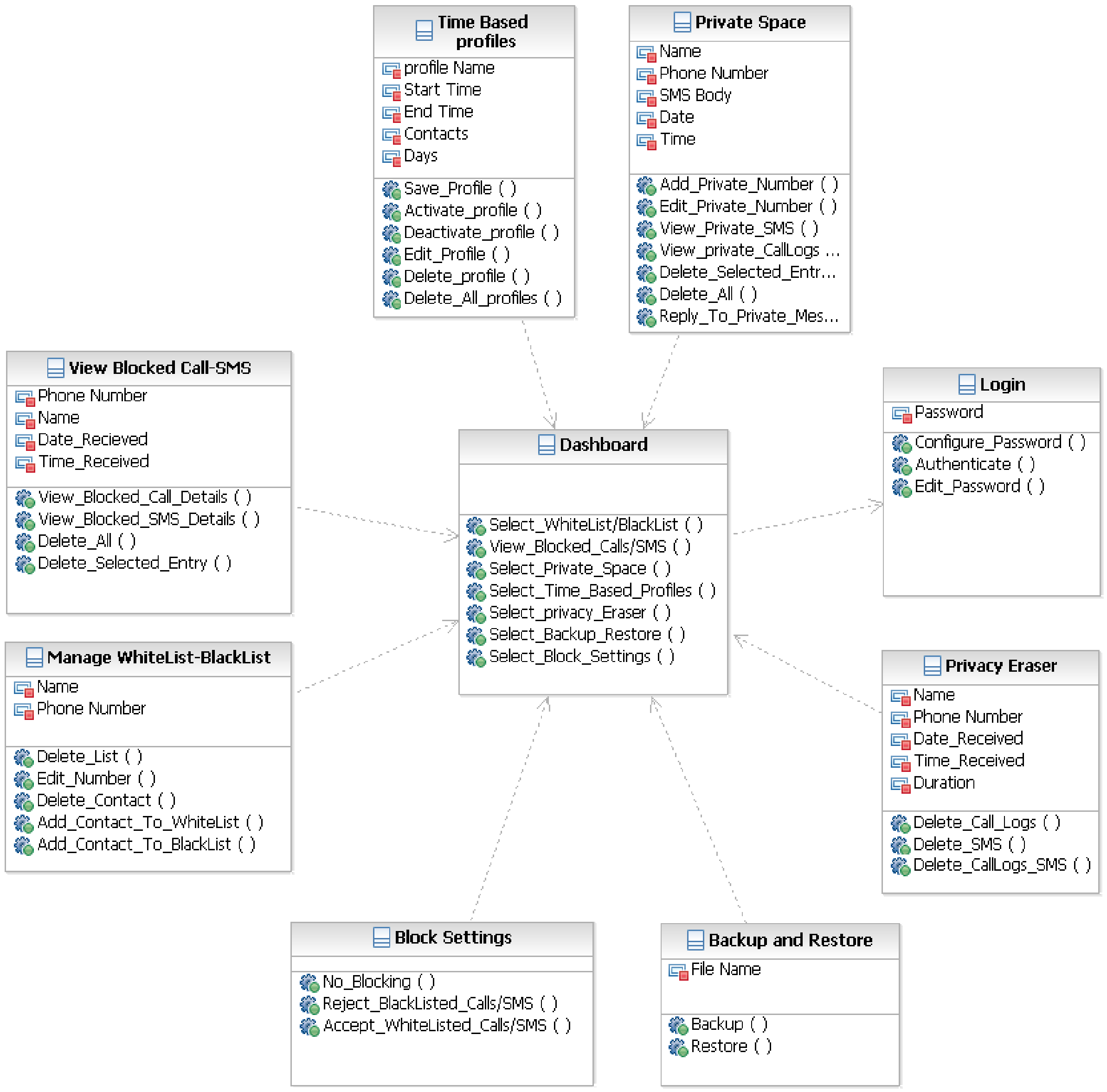


Figure 15: Class diagram

4.3.5. Deployment Diagram

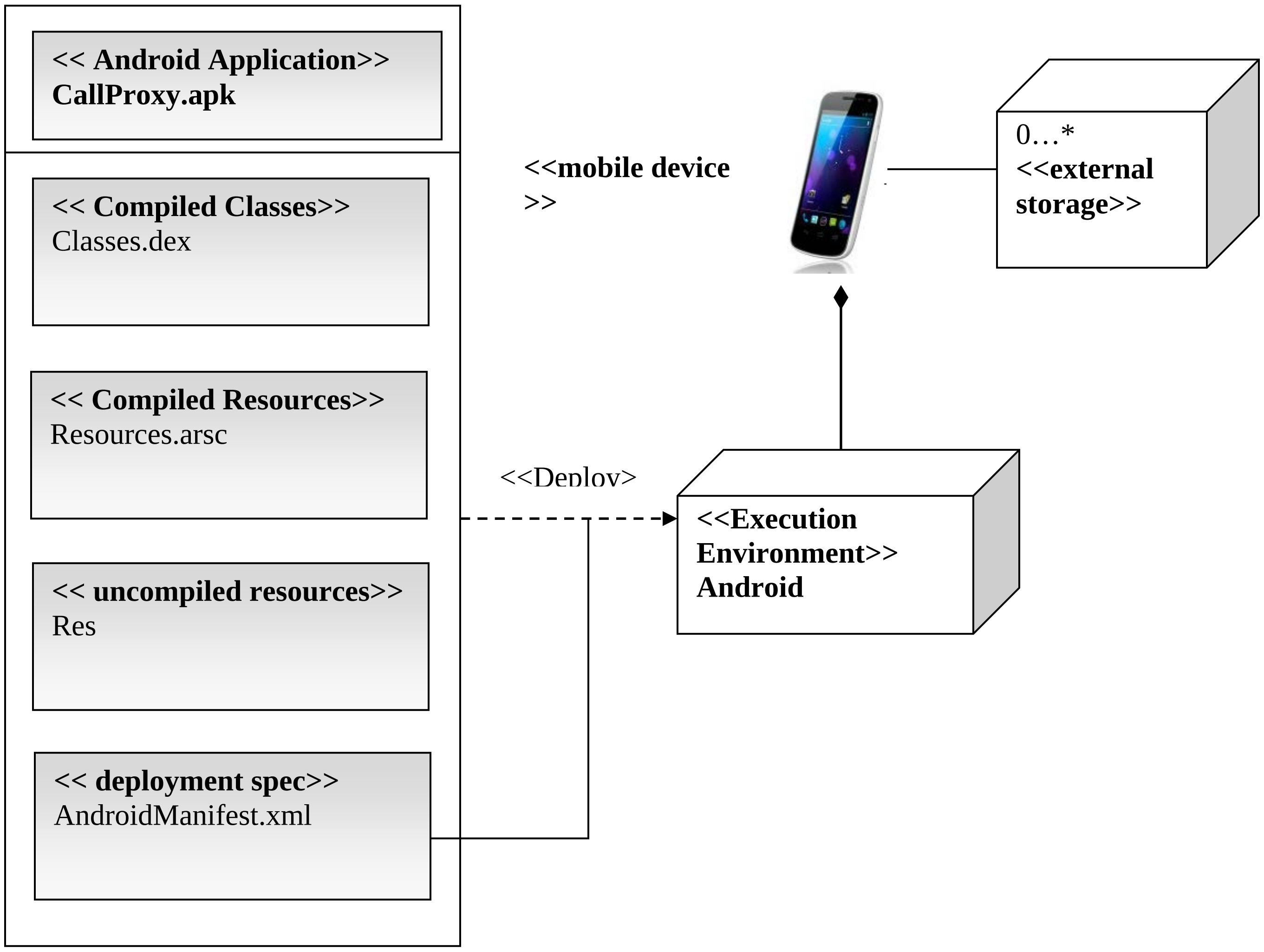


Figure 16: Deployment Diagram

4.4. High Level Design

4.4.1. Module Tree

Call Proxy

+ Login

- > LogDOA.java
- > LogDBHelper.java
- > LoginActivity.java

+ Dashboard

- > DashBoardActivity.java
- > Constants.java

→ Manage Black List

- > BLContactDisp.java
- > BLCallLogDisp.java
- > BInputNumber.java
- > BListDAO.java
- > BLDBHelper.java
- > Blacklist.java
- > BMyAdapter.java
- > BData.java
- > EditContact.java

→ Manage White List

- > WLContactDisp.java
- > WLCallLogDisp.java
- > WInputNumber.java
- > WListDAO.java
- > WLDBHelper.java
- > Whitelist.java

→ View Blocked call

- > CallBlock.java
- > BlockCallActivity.java

→ View Blocked SMS

- > CallBlock.java
- > ViewMsgDetails.java
- > MsgDBHelper.java

> ManageBlockedMsgActivity.java

→ Private Space
> PSContactDisp.java
> PSCallLogDisp.java
> PSInputNumber.java
> PSConfigure.java
> PSDAO.java
> PSDBHelper.java
> PSMGSDisp.java
> SMSReplyActivity.java
> ViewPMsgActivity.java

→ Settings
> SettingsActivity.java

→ Privacy Eraser
> PrivacyEraserActivity.java

→ Time Based Profiles
> Main.java
> TInsContact.java
> TInsCallLog.java
> TInputNumber.java
> ProfileDetails.java
> ProfileDisplay.java
> ProfileSettings.java
> TBDispContacts.java
> TBPDOA.java
> TBHelper.java

→ Back up
> Backup.java

→ Restore
> Backup_Restore_Activity.java
> Restore.java

4.4.2. Test Design

4.4.2.1. Introduction

The software test plan (STP) is designed to prescribe the scope, approach, resources, and schedule of all testing activities.

4.4.2.2. Objectives

Call Proxy is a call manager application. It will be tested for its efficiency in terms of blocking call or SMS.

4.4.2.3. Testing Strategies

Testing is the process of analyzing a software item to detect the differences between existing and required conditions and to evaluate the features of the software item.

Test plan components include

- Purpose of testing

  To check the functionality of the system.
- Items to be tested
  - Interface
  - Code
- Features to be tested
  - Reliability
  - Security
  - Performance
- Management and technical approach
  - System will undergo behavioral and functional testing first.
  - Manual testing will be done

- Pass/ fail criterion
    - Check on which input system fails and for which inputs it performs well.

- Risk assumption and constrains

4.4.2.4. Scope

Testing will be performed at several points in the life cycle as the product is constructed. Testing is an extremely dependent activity. As a result, test planning is a continuous activity performed throughout the system development life cycle.

4.4.2.5. Reference Material

- Project synopsis
- Concept document
- Software Requirement Specification
- Relevant UML diagrams

4.4.2.6. Test Items

- User Interfaces
- Code

4.4.2.7. Features to be tested

To check the reliability and security of the software it will be tested across different inputs. The system should not fail for any kind of input that is given to it. All the data that is entered into the system and all the data that is available in the system should be properly processed upon. All the reports should be correctly generated.

4.4.2.8. Approaches

The testing will start with the unit testing. here both white box and black box testing will be done. This process will take 3 days. Once unit testing is finished integration testing will be done in 3 days. The black box testing will be done using Rational Robot and white box testing will be done manually.

4.4.2.9. Component testing

This is also called unit testing. Here, testing will include testing of individual units like command buttons, Input boxes, List Views etc. Command buttons are tested for proper functionality. Input boxes were tested to check whether it allows the user to enter input from the keyboard or not. Also, the updates being made to the database were also tested.

4.4.2.10. Integration testing

Individual modules were integrated and the integrated modules were tested to check whether they perform their functions appropriately.

4.4.2.11. Interface testing

The application interface was tested to check whether it receives and interprets the required commands properly.

4.4.2.12. Performance testing

In this type of testing the performance of the system will be checked. This means that the response time of the system should not be less; i.e. The system should accept or Reject calls and SMS according to the appropriate active list.

4.4.2.13. Beta testing

System was tested by the end user and checked for the defects and the faults. Faults encountered, were fixed within 3 days.

4.4.2.14. Pass/fail criteria

In case of unit testing and integration testing, if the system gives the result same as expected result item has passed the test case.

If the test case fails to achieve expected result, then the test cases are ‘fail’.

4.4.2.15. Testing process

Testing deliverables

Here for unit testing, each and every unit will be tested for different inputs. This will be stored in Microsoft excel sheet with the columns as units on which testing is done, pre-requisite, objective, steps and data, expected result, actual result, pass/fail.

In case of integration testing, the fields of excel sheet are pre-requisite, test case description, steps and data, expected result, actual result, pass/fail.

Testing task

For starting with testing task, the system must be properly installed.

Resources

For testing purpose different resources like hardware and software resources as specified in software requirement specification will be given.

4.4.2.16. Environmental requirements

- **Software requirement**
  - Microsoft excel spreadsheet for documentation
  - Rational Robot software.

- **Risk and assumption**

  It was assumed that the testing process will follow the schedule as planned.

- **Change management procedure**

  If the process is getting delayed, the changes will be approved by the respective faculty and accordingly further testing process will be changed.

5. Implementation

5.1. Code

5.1.1. Block Call

```
package callproxy.dashboard;
import java.lang.reflect.Method;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import android.content.BroadcastReceiver;
import android.content.ContentValues;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.net.Uri;
import android.os.Bundle;
import android.telephony.TelephonyManager;
import android.util.Log;
import android.widget.Toast;
import com.android.internal.telephony.ITelephony;

public class CallBlock extends BroadcastReceiver {
   private ITelephony telephonyService;
   BListDAO dao = null;
   PSDAO dao1 = null;
   TBPDAO dao2 = null;
   SharedPreferences sp;
   TelephonyManager telephony;
   Context c1;

   @Override
   public void onReceive(Context ctx, Intent intent) {
          c1 = ctx;
          dao = new BListDAO(ctx);
          dao.open();
          dao1 = new PSDAO(ctx);
          dao1.open();
          sp = ctx.getSharedPreferences("timebased", 0);
          Log.i("Database Access", "Data...");
          telephony = (TelephonyManager) ctx
                      .getSystemService(Context.TELEPHONY_SERVICE);
          Bundle b = intent.getExtras();
          if (b != null) {
```

```
                        String state = b.getString(TelephonyManager.EXTRA_STATE);
                        String number =
b.getString(TelephonyManager.EXTRA_INCOMING_NUMBER);
                        if (state.equals(TelephonyManager.EXTRA_STATE_RINGING)) {
                                Log.i("MyBroadCastReceiver", number);
                                if (number.startsWith("+")) {
                                        number = number.substring(3, number.length());
                                        number = "0".concat(number);
                                }
                                boolean wflag = false;
                                boolean bflag = true;
                                int cur = sp.getInt("CurrentActivation", 0);
                                switch (cur) {
                                case 1:// timebased is active
                                        if (!isFromCurrentProfile(number))
                                        {
                                                blockCall();
                                                delLog(number);
                                                handleBlockedCall(number);
                                        }
                                        break;
                                case 2:// blacklist is active
                                        bflag= isBlackListed(number);
                                        if (bflag) {
                                                blockCall();
                                                delLog(number);
                                                handleBlockedCall(number);
                                        }
                                        break;
                                case 3:// whitelist is active
                                        wflag = isWhiteListed(number);
                                        if(!wflag)
                                        {
                                                blockCall();
                                                handleBlockedCall(number);
                                        }
                                        break;

                                default:
                                        break;

                                }
                                if (wflag == true || bflag == false) {
                                        isPrivate(number);
```

```
                    }
                }

            }
            dao.close();
            dao1.close();
        }
        private void delLog(String number) {
            try {
                Thread.sleep(2000);

                String strUriCalls = "content://call_log/calls";
                Uri UriCalls = Uri.parse(strUriCalls);

                int i = c1.getContentResolver().delete(UriCalls,
                        "NUMBER = '" + number + "'", null);
            } catch (Exception e) {

            }
        }
        private void blockCall() {
            try {
                Class c = Class.forName(telephony.getClass().getName());
                Method m = c.getDeclaredMethod("getITelephony");

                m.setAccessible(true);
                telephonyService = (ITelephony) m.invoke(telephony);
                telephonyService.endCall();
                Toast.makeText(c1, "Call Rejected", Toast.LENGTH_LONG).show();
            } catch (Exception e) {

            }
        }
        private boolean isFromCurrentProfile(String number) {
            boolean flag = false;
            String[] params = { sp.getString("CurrentProfile", "") };
            dao2 = new TBPDAO(c1);
            dao2.open();

            Cursor cursor = dao2.getProfileDetails(params);
            if (cursor.moveToNext()) {
                String ltname = cursor.getString(3);
                SQLiteDatabase db1 = c1.openOrCreateDatabase("Db.db",
                        Context.MODE_WORLD_WRITEABLE, null);
                        String[] parms = { number };
                        Cursor result = db1.query(ltname, new String[] { "name", "phno" },
                                        "phno" + "=?", parms, null, null, null);
                        if (result.moveToNext())
                                flag = true;

                        result.close();
                        db1.close();
                }
                cursor.close();
                dao2.close();

                return flag;
        }
        private boolean isBlackListed(String number) {
                boolean flag = false;
                String[] params = { number };
                Cursor cursor = dao.getBlistNo(params);
                if (cursor.moveToNext()) {
                        flag = true;
                }
                cursor.close();
                return flag;
        }
        private boolean isWhiteListed(String number) {
                boolean flag = false;
                String[] params = { number };
                Cursor cursor = dao.getWlistNo(params);
                if (cursor.moveToNext()) {
                        flag = true;
                }
                cursor.close();
                return flag;
        }
        private void handleBlockedCall(String number) {

                Calendar cal = Calendar.getInstance();
                SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
                String formattedDate = df.format(cal.getTime());

                ContentValues cv = new ContentValues();
                cv.put(Constants.BLNAME, "");
                cv.put(Constants.BLPH, number);
```

```
            cv.put(Constants.BLDTIME, formattedDate);
            long id = dao.insertInLog(cv);
            if (id == -1) {
                    Toast.makeText(c1, "Number could not be added",
Toast.LENGTH_LONG)
                                    .show();
            }
   }
   private void isPrivate(String number) {
            Cursor cursor2 = dao1.getPCNo(new String[] { number });
            if (cursor2.moveToNext()) {
                    Calendar cal = Calendar.getInstance();
                    SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
                    String formattedDate = df.format(cal.getTime());
                    ContentValues cv = new ContentValues();
                    String name = dao1.getPCName(new String[] { number });
                    cv.put(Constants.PNAME, name);
                    cv.put(Constants.PPH, number);
                    cv.put(Constants.PDTIME, formattedDate);
                    long id = dao1.insertInPLog(cv);
                    if (id == -1) {
                            Toast.makeText(c1, "Number could not be added",
                                            Toast.LENGTH_SHORT).show();
                    }
            }
            cursor2.close();
   }
}
```

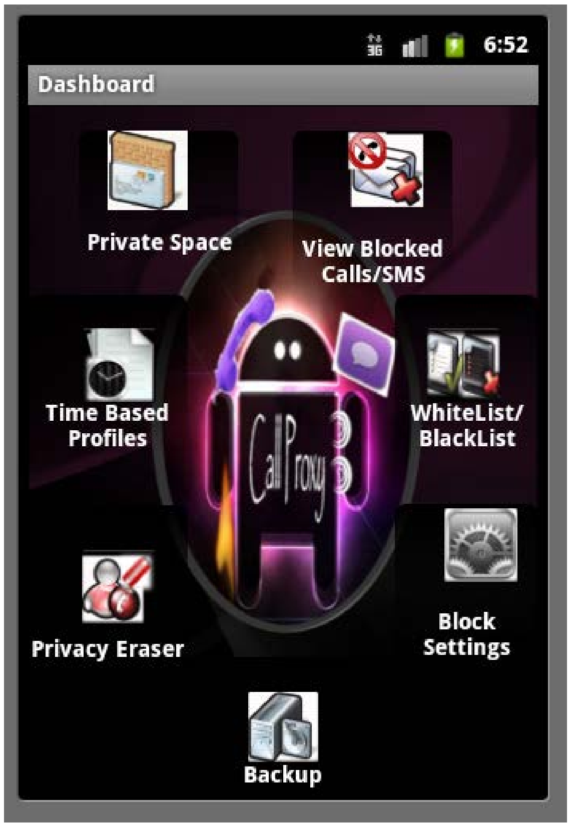
5.1.2. Profile Service

```
package callproxy.dashboard;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.Timer;
import java.util.TimerTask;
import android.app.Service;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Handler;
import android.os.IBinder;
import android.widget.Toast;

public class ProfileService extends Service {

	private TBPDAO dao;
	@Override
	public void onCreate() {
	}
	@Override
	public int onStartCommand(Intent intent, int flags, int startId) {
		check();
		TimerTask scanTask;
		final Handler handler = new Handler();
		Timer t = new Timer();
		scanTask = new TimerTask() {
			public void run() {
				handler.post(new Runnable() {
					public void run() {
						check();
					}
				});
			}
		};
		t.schedule(scanTask, 0, 60000);
		// We want this service to continue running until it is explicitly
		// stopped, so return sticky.
		return START_STICKY;
	}
	@Override
	public IBinder onBind(Intent arg0) {
		// TODO Auto-generated method stub
		return null;
	}
	public void check() {
		Context context = ProfileService.this;
		dao = new TBPDAO(context);
		dao.open();

		Calendar cal = Calendar.getInstance();
		int minutes = cal.get(Calendar.MINUTE);
		int hours = cal.get(Calendar.HOUR);
		int AM_orPM = cal.get(Calendar.AM_PM);

		if (AM_orPM == 1) {
			hours += 12;
		}
		SimpleDateFormat sdf = new SimpleDateFormat("EEEE");
		Date d = new Date();
		String dayOfTheWeek = sdf.format(d);

		String ctime = hours + ":" + minutes;
		Cursor cursor = dao.getAllProfiles();
		while (cursor.moveToNext()) {
			if (ctime.equals(cursor.getString(1))&&
"A".equalsIgnoreCase(cursor.getString(5))) {
				Toast.makeText(ProfileService.this,"Hiii..",
						Toast.LENGTH_SHORT).show();
				SQLiteDatabase myDataBase = null;
				myDataBase = ervice.this.openOrCreateDatabase("Db.db",
						Context.MODE_WORLD_WRITEABLE, null);
				Cursor c = myDataBase.query(cursor.getString(4),
						new String[] { "days" }, "days=?",
						new String[] { dayOfTheWeek }, null, null,
null);
				if (c.moveToNext()) {

					SharedPreferences sp =
edPreferences("timebased", 0);
					SharedPreferences.Editor editor = sp.edit();
					editor.putString("CurrentProfile",
cursor.getString(0));
```

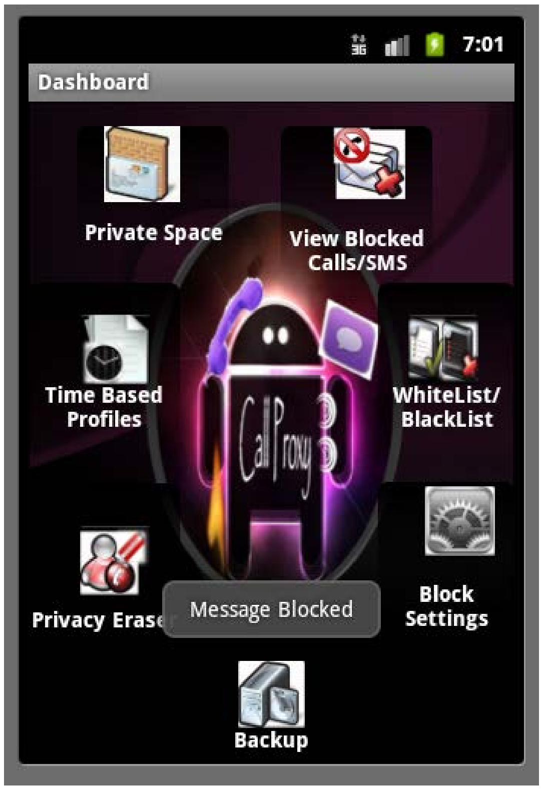
```
                                        int cur = sp.getInt("CurrentActivation", 0);
                                        editor.putInt("PreActivation", cur);
                                        editor.putInt("CurrentActivation", 1);
                                        editor.commit();
                                        editor.commit();
                                        Toast.makeText(
                                                ProfileService.this,
                                                sp.getString("CurrentProfile", "")
                                                        + " Profile
Activated..!",
                                                Toast.LENGTH_SHORT).show();
                                }
                                c.close();
                                myDataBase.close();
                        }
                        if (ctime.equals(cursor.getString(2)) &&
"A".equalsIgnoreCase(cursor.getString(5))) {
                                SharedPreferences sp =
getSharedPreferences("timebased", 0);
                                SharedPreferences.Editor editor = sp.edit();
                                editor.putString("CurrentProfile", "NOPROFILE");
                                int pre = sp.getInt("PreActivation", 0);
                                editor.putInt("CurrentActivation", pre);
                                editor.putInt("PreActivation", 0);
                                editor.commit();
                                Toast.makeText(ProfileService.this,
                                        cursor.getString(0) + " Profile
Deactivated..!",
                                        Toast.LENGTH_SHORT).show();
                        }
                }
                cursor.close();
                dao.close();
        }

}
```

5.2. Screen Shots

5.2.1. Dashboard

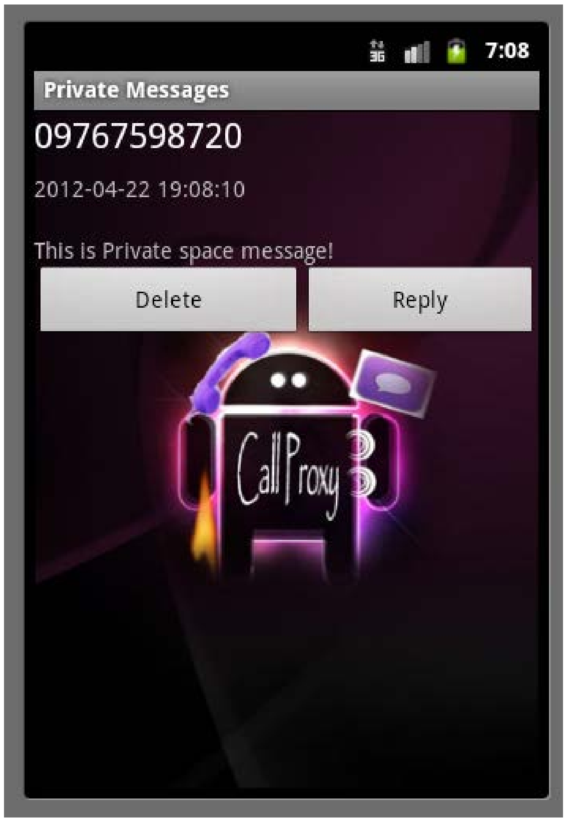


5.2.2. Blocked SMS



Message Blocked

5.2.3. Private Space

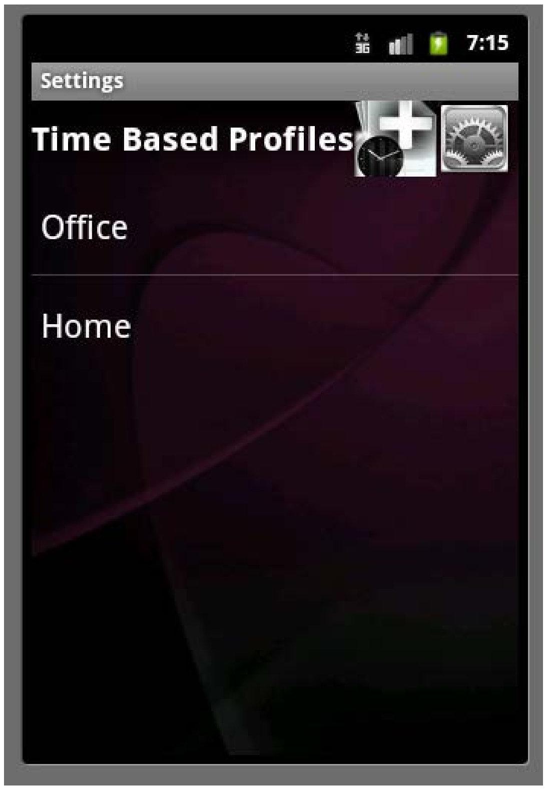


09767598720

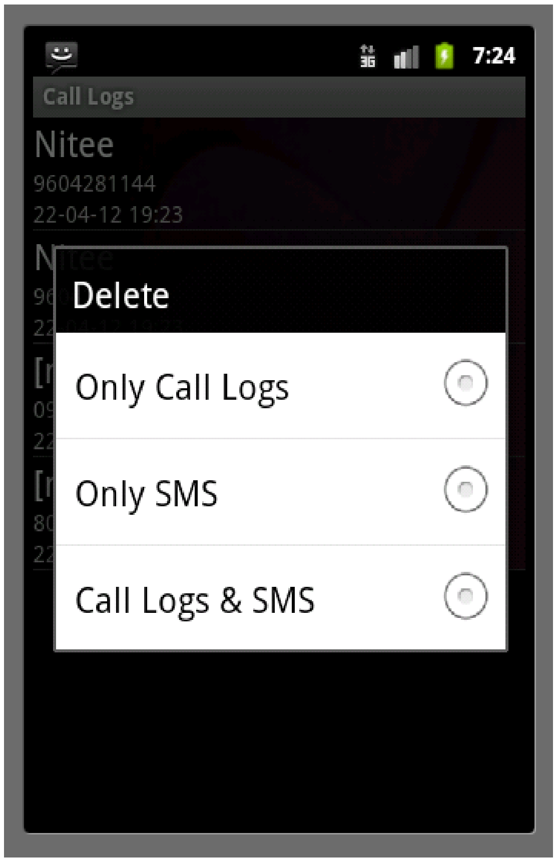
2012-04-22 19:08:10

This is Private space message!

5.2.4. Time Based Profile



5.2.5. Privacy Eraser



5.3. Testing

Test Cases

Black box Testing

Sr #	Field Name	Objective	Steps & Data	Expected result	Actual Result	Pass/ Fail
1 Login						
1.1	Ok	To accept password and authenticate user.	Enter the password and click on Ok button	Access Should be granted. On wrong password Access should be denied.	Authenticated!	Pass
1.2	Cancel	Login process should be cancelled.	Click on Cancel button	Application should be closed.	Login Cancelled!	Pass
2 Dashboard						
2.1	Private Space	To save personal conversation	Click on Private Space Button	It should open the private space module.	Private space opened for authentication	Pass
2.2	View Blocked Calls/SMS	To display all logs and messages from blocked numbers.	Click on View Blocked Calls/SMS Button	It should display blocked calls n messages.	Blocked Call logs/ SMS are displayed	Pass

2.3	White List/ black List	To add contacts to black list and white list	click on White List/ black List button	White List/ Black List configuration module should be displayed.	Configuration module is displayed	Pass
2.4	Time Based Profiles	To activate time based profile and allow specific calls during activation period of the profile	click on Time Based Profiles button	It should open time based profile module	Module is opened	Pass
2.5	Privacy Eraser	To delete call and SMS log/logs entry/entries of specific number.	click on Privacy Eraser button	It should open Privacy Eraser module and should display recent logs.	Module is opened also recent logs are displayed	Pass
2.6	Backup	To back up application data.	click on Backup button	It should open the module and ask for retrieval/Backup	Module opened successfuly	Pass
2.1 Private Space						
2.1.1	Add Private Number	To add new contact in private space	Click on Add Private Number	Private Number and Name should be added. Also the Private space contact list should be updated.	Number added successfully and contact list is refreshed.	Pass

2.1.2	View Private Call logs	All the call logs from private number should be displayed here not in phone call logs.	click on View Private Call logs button	All recent call logs from private number should be displayed along with date and time.	Private Call logs are displayed successfully	Pass
2.1.3	View Private Messages	All the Private Messages should be displayed as well as reply should be maintained from this module not through phone's messaging module	click on View Private Messages button	All recent messages from private number should be displayed along with date, time and body of the message. Reply facility should be provided.	Private messages displayed successfully. Reply to the private message is done	Pass
2.2 View Blocked Calls/ SMS						
2.2.1	View Blocked Calls	To display all blocked call logs	Click on View Blocked Calls Button	All recent blocked call logs should be displayed. On long click option Delete operations should be performed.	Blocked call list is displayed	pass
2.2.2	View Blocked SMS	To display all blocked SMS logs as well as body of message.	Click on View Blocked SMS Button	All blocked SMS should be displayed along with their logs. On long click option Delete operations should be performed.	SMS logs as well as its body is displayed on its click	Pass
2.3 Manage White List/ Black List						
2.3.1	Manage White List	To configure white list	Click on Manage White list button	New contacts should be added or Added white list contacts should be displayed. On long click option Delete/Edit operations should be performed.	New contact added successfully also white list is displayed. Delete/Edit operation is working properly.	Pass
2.3.2	Manage Black List	To configure Black list	Click on Manage Black list button	New contacts should be added or Added Black list contacts should be displayed. On long click option Delete/Edit operations should be performed.	New contact added successfully also Black list is displayed. Delete/Edit operation is	Pass

					working properly.	
2.3.3	Settings	To activate or deactivate White list/ Black List	Click on Settings Button	A dialog should be displayed with activation deactivation list options	Lists activated/ deactivated successfully.	Pass
			2.4 Time Based Profiles			
2.4.1	Time Based Profiles	To display all the saved profiles	click on Time Based Profiles Button	Saved profiles should be displayed or else "no profiles created" notification should be given.	All saved profiles are displayed	Pass
2.4.2	Add new profile	To create new profiles	click on add new button	Should open a form to fill up new profile details and should save the profile.	New profile is created successfully.	Pass
2.4.2.1	Profile name	To give name to profile	Enter the profile name in text field	Should not accept the repeated profile name.	New name is accepted. On repeated name Error message is displayed.	Pass
2.4.2.2	Start time	To give the profile activation time.	Click on set button and give the timing	Should not accept if other profile is created with same time	Start time is accepted. On repeated timings Error message is displayed.	Pass
2.4.2.3	End Time	To give the profile deactivation time.	Click on set button and give the timing	Should not accept if start time and end time is same.	End time is accepted. On repeated timings Error message is displayed	Pass

2.4.2.4	Insert Contacts	To provide contacts for specific profile so that calls from those number will be accepted if that profile is activated in given time.	Click on Insert contacts button and select option category by clicking on menu option	Contacts should be retrieved based on the category. On long press contacts should be added to profile. Should not accept duplicate entry.	Contacts added successfully. On same entry error message is displayed.	Pass
2.4.2.5	Insert days	To provide days for specific profile so that profile will be activated for that day.	Click on Insert days Button and select days	Days should be added to profile. Default all days should be selected.	Days are added successfully	Pass
2.4.2.6	Save	To save the profile	Click on save button	Newly created profile should be saved.	Profile saved successfully	Pass
2.4.2.7	Cancel	To cancel the add new profile module	Click on Cancel Button	Form should be cancelled.	Profile form is cancelled	Pass
2.4.3	Settings	To Edit existing profile	Click on settings button	Edit profile form should be displayed for selected profile. Profile should be saved.	Profile edited successfully	Pass
2.5 Privacy eraser						
2.5.1	Privacy Eraser	To delete call and SMS log/logs entry/entries of specific number	Click on the module	Call logs should be retrieved and on long press it should delete call logs/ SMS of a particular number	Call logs /SMS deleted successfully	Pass

2.6 Backup						
2.6.1	Backup	To backup application data	Click on Backup Button	Backup should be taken on SD card	Backup successful	Pass
2.6.2	Restore	To restore backup data	click on Restore button	Data should be restored in previous format	Restored successfully	Pass

6. Effort Estimation

6.1. Cost Estimation Method

Software cost estimation is the process of predicting the amount of effort required to build software system. Models provide one or more mathematical algorithms that compute cost as a function of a number of variables. Size is a primary cost factor in most models. It can be measured using:

- Lines of Code
- Function Point

Lines of Code

The most commonly used measure of source code program length is the number of lines of code. The abbreviation NCLOC is used to represent a non –commented source line of code. NCLOC is also sometimes referred as effective lines of code (ELOC). NCLOC is therefore a measure of the uncommented length.

The commented length is also valid measure, depending on whether or not line Documentation is considered to be a part of programming effort. The abbreviation CLOC is used to represent a commented source line of code. KLOC is used to denote thousands of lines of code.

Function Points

Function points measure size in terms of the amount of functionality in a system. Function points are computed by first calculating an unadjusted function point count (UFC). Counts are made for the following categories.

- External Inputs- Those items provide by the user that describe distinct application oriented data (such as file names and menu selections)
- External Outputs-Those items provided to the user that describe distinct application oriented data (such as reports and messages, rather than the individual components of these)

- External inquiries- Interactive inputs requiring response
- External files-machine readable interface to other systems.
- Internal files- logical master files in the system

Once this data has been collected, a complexity rating is associated with each count. Each count is multiplied by its corresponding complexity weight and the results are multiplying the UFC by technical complexity factor (TCF).

6.2. Types of models

There are two types of models that have been used to estimate cost:

- Cost Models
- Constraint Models

6.3. Cost Estimation

Measurement parameter	Weighting factor			
	Simple	Average	Complex	Sum
Number of user inputs	3*3	4*9	6*4	
Number of user outputs	4*23	5*19	7*9	
Number of user inquiries	3*0	4*2	6*5	
Number of files	7*1	10*4	15*1	
Number of external interfaces	5*0	7*0	10*3	
Total(∑ FP)				449

<u>Complexity factor assumptıons</u>

Sr. No.	Factor	Value
1	Back-up and Recovery ?	5
2	Data Communication ?	3
3	Distributed Processing ?	0
4	Performance Critical ?	3
5	Existing Operational Environment ?	3
6	On-line Data Entry ?	2
7	Input transactions over multiple Screens?	2

8	Online Master File Updates ?	2
9	Information Domain Values Complex ?	1
10	Internal Processing Complex?	4
11	Code Designed for reuse?	2
12	Conversion / installation in Design?	4
13	Multiple Installations?	1
14	Application Designed for change?	2
	∑ (Fi)	**34**

<u>Estimated Number of Adjusted FP is derived using the following Formula</u>:-

FP *ESTIMATED* = (FP COUNT TOTAL * [COMPLEXITY ADJUSTMENT FACTOR])

Complexity Adjustment Factor is [0.65+ (0.01*∑(Fi)
Complexity Adjustment Factor = [0.65 + (0.01 * 34)] = 0.99

ESTIMATED FP COUNT = (449* 0.99) = 444.51

<u>Assume :</u>

- Productivity =**14 FP/Month**
- Cost = **Rs. 3000/Month**

<u>Cost for a Function Point (COST / FP) is calculated as follows</u>

- **Cost / FP = (3000 /14)**
  = Rs. 214.28 (Cost for 1 FP)

<u>Total Estimated Project Cost and Project Effort can be calculated as</u>:-

- **Total Estimated Project Effort = (449/ 60) = ~ 7.4 Man Months**

7. Schedule of Work

Month	Work done	Done by
September	Basic User Interface	Tejashree, Sayali, Shruti, Nitee
October	Call Proxy Presentation (PPT)	Tejashree, Shruti
	Configuration Flowchart	Sayali
	Software Requirement Specification (SRS) Document	Shruti
	Data Flow Diagram	Nitee
	System Flowchart	
	Use-case diagrams	
December : 7th DEC, 2011	I -Semester External Project Evaluation	
December	Login feature for the application	Tejashree
	Project Logo Designing	Sayali
	List view UI for application content	
	Deprecated ITelephony interface	Shruti
	Call blocking from blacklisted number	Nitee
	SMS blocking from blacklisted number	
	Creating 'Broadcast Receiver' to block calls and messages	
	Storing blocked call logs and messages in the database	
January	Creation of private space and Storing call logs and messages of private contacts and displaying them	Nitee, Tejashree
	Retrieving Incoming message details : message body, date and time and storing in the database for Private Space module	Nitee, Tejashree
	Applying Background to each activity	Tejashree
	Adding number to the White List/Black List from the Phone Call logs, Contacts and Input Number	Shruti
	Deleting specific call logs and messages from Private Space module.	Nitee
	Creation of Time based profiles, providing choice to add contacts to every profile from Call logs, Contacts and Input Number. Also providing choice of days for every profile	
	Project validation: Exact 10 (mobile number) /11 (landline number) /14 (Both numbers including country code) digit number for Input Number Choice	

Month	Work done	Done by
February	Creating and Maintaining Dynamic Database in SQLite to allow user to create n number of profiles with any number of contacts in it.	Nitee, Tejashree
	Erasing Call Logs and SMS of selected number from Inbox along with updating the inbox data.	Tejashree, Sayali
	Assigning Icons to each Module	Tejashree
	Taking Backup of the application data to the SD Card	Shruti
	Display of database contents using a two-line as well as a three line inflated list view.	
	Creation of Time based profiles, providing choice to add contacts to every profile from Call logs, Contacts and Input Number. Also providing choice of days for every profile	Nitee
	Creating a Service to Activate and deactivate a profile and keeping track of the Activated profile	
March: 7th MAR 2012	1st Internal Project Evaluation	
March	Giving Animated transitions for each module	Tejashree
	Providing Reply for private space messages	Sayali
	Restoring backed up data from SD Card to Application Database	Shruti
	Deleting specific call logs and messages from Private Space module	
	Deleting specific call logs and messages from view blocked calls and messages module	
	Providing the choice to Edit and Delete a particular profile	Nitee
	Providing the choice to Activate and Deactivate a profile manually	
	Standardizing phone number of the incoming call/message. i.e. considering the number with country code (+91) or Zero (0) or simply a 10/11 digit phone number in Broadcast Receiver	

Month	**Work done**	**Done by**
April	Unit Testing for each module	Tejashree
	Full fledge, user friendly Interface for whole project	
	Calculating Function Point Analysis for Effort Estimation	Sayali
	Standardizing the application user interface by using a standard interaction style on every activity	Shruti
	Project validation: Sorting retrieved contacts alphabetically	
	Refreshing activity data	
	Managing activation/deactivation of black list/ white list, time based profiles through shared preferences	Nitee
	Managing (pre) activation of black list/white list as soon as a particular time based profile exits by handling shared preferences	
	Project validation: Refreshing the activity data and Finishing activity wherever needed	
April: 7th APR 2012	2nd Internal Project Evaluation	
April	Final Project Report	Tejashree, Sayali, Shruti, Nitee

8. Conclusion and Future Scope

A novel feature – *Fake Space Creator* will be added to the Call Proxy. Fake Space will create a fake space to protect the privacy of contents of the phone. On the failure of authentication at the time of unlocking the phone, the app will activate fake space, which will deny access to all the data on the phone, by showing mock data.

A new feature - *Location Based reminder system* (LBRS) can be added to the project. It will help users to co-ordinate with their day-to-day activities with their demanding schedules. It reminds will them of the activities they need to perform at particular locations. This feature will use GPS to locate the cell phone, periodically check the location and verify it with the database checking for reminders set on that particular location.

As per the literature survey, there is no such application which provides the location of the calling party onto the home screen of the user. Applications which provide this feature, do so with the use of GPS (Global Positioning System).

References:

- **Software**

[1] http://www.talkandroid.com/google-android-application-guide/

[2] http://thetwistedcables.com/android/how-to-start-android-development-a-startup-guide-for-workspace-installation/

[3] http://mobiforge.com/developing/story/getting-started-with-android-development

- **World Wide Web**

[1]https://www.market.android.com/details?id=com.netqin.mm&feature=search_resul

[2]https://www.market.android.com/details?id=it.mandave.prefix&feature=search_result

[3] https://www.market.android.com/search?q=call+manager&so=1&c=apps

[4]https://www.market.android.com/details?id=com.mrnumber.blocker&feature=search_result

[5] https://www.market.android.com/?hl=en