Collection of Smartphone Functionalities: An Android Application

GRADUATE PROJECT

Submitted to the Faculty of
The School of Engineering & Computing Sciences
Texas A&M University-Corpus Christi
Corpus Christi, TX

In Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

by

Rajesh Kumar Srirangam
Fall 2013

Committee Members

Dr. Ajay Katangur
Committee Chairperson

Dr. Ahmed Mahdy
Committee Member

Dr. David Thomas
Committee Member
ABSTRACT

The demand for mobile devices is increasing as more people rely on smartphones to stay connected and be productive while away from personal computers. Different operating systems such as Android, iOS, Symbian and Windows provide numerous applications, which provides various services on smartphones. Out of all these operating systems Android tops the smartphone market with almost 70% share due to its open nature that encourages developer community to develop various applications by using the open–source code [13]. This project presents a new application that provides all the smartphone functionalities such as Search, Bluetooth, Wi-Fi, Calculator, Screen Lock and Flashlight in a single application. These are the most commonly used functionalities on smart phones and a collection of these functionalities in a single application provides faster and better usage. It eliminates the user effort to move between various screens to access these functionalities.
# TABLE OF CONTENTS

Abstract .................................................................................................................................................. ii

Table of Contents .................................................................................................................................... iii

List of Figures .......................................................................................................................................... v

1.  Background and Rationale ................................................................................................................ 1
   1.1 What is Android? ............................................................................................................................ 2
       1.1.1 Eclipse Integrated Development Environment ................................................................. 2
       1.1.2 Android Software Development Kit ..................................................................................... 3
       1.1.3 Android Development Plugins Tools for Eclipse ............................................................... 3
   1.2 Motivation ......................................................................................................................................... 3

2.  Requirements and Application design ............................................................................................... 5
   2.1 Requirements .................................................................................................................................... 5
   2.2 Application Design .......................................................................................................................... 5
       2.2.1 Layouts ...................................................................................................................................... 5
       2.2.2 Activities ................................................................................................................................... 7
       2.2.3 Classes ........................................................................................................................................ 9

3.  Implementation of Modules ................................................................................................................ 10
   3.1 Search Module ................................................................................................................................. 10
       3.1.1 Contact Search ......................................................................................................................... 11
       3.1.2 Music Search ........................................................................................................................... 13
       3.1.3 Message Search ........................................................................................................................ 14
       3.1.4 Application Search ................................................................................................................... 15
       3.1.5 File Search ............................................................................................................................... 16
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1.6</td>
<td>Global Search</td>
<td>17</td>
</tr>
<tr>
<td>3.2</td>
<td>Collection Module</td>
<td>18</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Bluetooth</td>
<td>18</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Wi-Fi</td>
<td>19</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Calculator</td>
<td>20</td>
</tr>
<tr>
<td>3.2.4</td>
<td>Screen Lock</td>
<td>21</td>
</tr>
<tr>
<td>3.2.5</td>
<td>Flash Light</td>
<td>21</td>
</tr>
<tr>
<td>4.</td>
<td>Testing and Evaluation</td>
<td>23</td>
</tr>
<tr>
<td>4.1</td>
<td>Testing of Search Module</td>
<td>23</td>
</tr>
<tr>
<td>4.2</td>
<td>Testing of Collection Module</td>
<td>24</td>
</tr>
<tr>
<td>5.</td>
<td>Conclusion and Future Work</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>Bibliography and References</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>Appendix A</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>A.1 Performance Testing on Android Devices</td>
<td>36</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 1. Smartphone OS market Share .................................................................1
Figure 2. Home screen of the app ...........................................................................6
Figure 3. Layout of Collection Module .................................................................6
Figure 4. Layout of Result Entity ...........................................................................7
Figure 5. Layout of Result List ..............................................................................7
Figure 6. Code snippet of Contact Search ...........................................................12
Figure 7. Code snippet of Contact Search ...........................................................12
Figure 8. Code snippet of Music Search ..............................................................13
Figure 9. Code snippet of Music Search ..............................................................14
Figure 10. Code snippet of Message Search .......................................................15
Figure 11. Code snippet of Application Search ..................................................16
Figure 12. Code snippet of File Search ...............................................................17
Figure 13. Code snippet of the Bluetooth ............................................................19
Figure 14. Code snippet of Wi-Fi ..........................................................................19
Figure 15. Code snippet of Calculator ...............................................................20
Figure 16. Code snippet of Screen Lock .............................................................21
Figure 17. Code snippet of Flashlight .................................................................22
Figure 18. Testing of Contact Search .................................................................23
Figure 19. Testing of Music Search ......................................................................24
Figure 20. Testing of Message Search ...............................................................25
Figure 21. Testing of Application Search ............................................................26
Figure 22. Testing of File Search ..........................................................................27
Figure 23. Testing of Global Search.............................................................................28
Figure 24. Testing of Collection Module........................................................................29
Figure 25. Testing of Bluetooth.......................................................................................30
Figure 26. Testing of Wi-Fi .............................................................................................31
Figure 27. Testing of Calculator.......................................................................................32
Figure 28. Testing of Flashlight.......................................................................................33
1. BACKGROUND AND RATIONALE

Smartphone is a device that can provide various computational and communicational services. There is been a great improvement in the technology of smartphone devices in recent years. Numerous services are provided within these small devices, which are used in our day-to-day life. With increase in usage of smartphones, development of smartphone applications has also increased. Currently there are millions of applications, which are developed for various platforms such as Android, iOS, Symbian and Windows etc. Out of all these platforms Android occupies over 70% of the smartphone market [13] as shown in figure 1. It powers millions of mobile devices in more than 190 countries [1].

![Smartphone OS Market Share, 4Q 2012](forums.macrumors.com, 2013)

Figure 1: Smartphone OS market Share (forums.macrumors.com, 2013)
1.1 What is Android?

Android is a Linux-based operating system designed for smartphones and tablets. The user interface of android is based off touch inputs that corresponds to real world actions like swiping, tapping, pinching etc. to manipulate on-screen objects [15]. It is developed by Android Inc., which is acquired by Google in July 2005. The openness of the platform is one benefit of developing android apps. It was found in a survey conducted in April-May 2012 that it is the most popular platform for developers, used by 71% of mobile developer population [13]. There are currently more than 850,000 apps for android [14].

Android comes in various versions such as 1.x, 2.x, 3.x, 4.x etc. [15]. Most of the android apps are developed in Java, which is one of the most widely used programming languages. The Applications developed in Java runs on variety of devices, as Java is platform independent. The tools that are required to develop an android app in Java are as follows.

- Eclipse Integrated Development Environment (IDE)
- Android Software Development Kit (SDK)
- Android Development plugin Tools for Eclipse

1.1.1 Eclipse Integrated Development Environment

Eclipse is a Multi-Language Integrated Development Environment (IDE) that comprises plugin system and workspace that is used to develop various applications in different programming languages like C, C++, Java, COBAL, PHP etc. [16]. It is widely used to write Java applications. It includes various development tools that help the developer. It is free and open source software.
1.1.2 Android Software Development Kit

The process of developing new applications for Android is called Android Software Development. The applications are developed using Android software development kit that includes software development tools [16]. It comprises of debuggers, emulators, Application programming Interface (API) tools, tutorials, libraries and sample code. The android applications are developed in Java using android software development kit in Eclipse.

1.1.3 Android Development plugin Tools for Eclipse

The Android Development Tools (ADT) for Eclipse enables us to use the Android software development kit tools in the Eclipse Integrated Development Environment to develop applications in android platform. Android Development tools helps the developers to develop the projects quickly, add packages based on android framework and debug the code etc. The usage of Android development tools with Eclipse is one of the fastest ways to develop the applications in android platform.

1.2 Motivation

Basic smartphone functionalities such as Search, Bluetooth, Wi-Fi, Flashlight, calculator etc. are the most commonly used apps by users of a smart phone in day-to-day life. In general in order to access these functionalities the user needs to move to a specific location on the device. As these functionalities are frequently used it will be difficult for the user to move to a specific location and access them. This difficulty motivated me to develop an application that would eliminate such effort by bringing these functionalities to a single location and provide faster and better usage. This application will be
developed on android so that it reaches vast majority of the people as Android covers over 70% of smartphone market.
2. REQUIREMENTS AND APPLICATION DESIGN

2.1 Requirements

Millions of applications are developed and many more are being developed for smartphone users. These are developed for various platforms like Android, iOS, Symbian, and Windows etc. Each application has different purpose, provides specific services and reduces user effort in its own way. This project aims at providing a convenient way of accessing different smartphone functionalities. This application helps user to access these frequently used functionalities with less effort. This project is developed using the Android SDK version 3.0 (Honeycomb) as the lower version of Android does not contains some features of Bluetooth and Wi-Fi. Below are the tools that are used to develop this Android application.

- Eclipse Integrated Development Environment (IDE)
- Android Software Development Kit (SDK) 3.0 (Honeycomb)
- Android Development plugin Tools (ADT) for Eclipse
- Java SE 6 Development Kit (JDK)

2.2 Application Design

2.2.1 Layouts

Layouts are the user interface components that user can view and interact with the application. The Layouts that are used in this application are as follows:

Main.xml: The Figure 2 shows the layout, which is the home screen of the application. It contains Contact Search, Music Search, Message Search, Application Search, File.
Search, Global Search and Collection Module as buttons, which takes user to their specific screens on a single touch of a button.

Figure 2: The layout of the Home Screen

Activity_all_in_one.xml: With single touch of Collection module button on home screen, it takes user to this layout. It contains Bluetooth, Wi-Fi, Calculator, Screen Lock and Flashlight buttons that performs their respective actions on a single touch of a button as shown in figure 3.

Figure 3: Layout of Collection Module
List_row_layout.xml: This layout is the structure of entities that are returned by the search operation. It contains Title of the entity, Type and Image as shown in figure 4.

![Figure 4: Layout of Result Entity](image1)

Activity_search.xml: This layout contains the search view and list of the resultant entities that are returned by search operation and is shown in figure 5.

![Figure 5: Layout of Result List](image2)

2.2.2 Activities

The activities that are used in the application are as follows:

Mainactivity.java

This is the main activity of the application. This invokes the other activities such as ContactSearch, MusicSearch, MessageSearch, AppSearch, FileSearch, SearchActivity and AllInOneActivity.
ContactSearch.java

This activity performs the functionality of searching through contacts. When user gives input in the search box, it searches for the contacts that contain the user string and the results are displayed as a list on the user screen.

MusicSearch.java

This activity performs the functionality of searching through music. User inputs the string in the search box, it searches for the music files in internal and external memory (SD Card) of the device that contains the user string in its title, and the results are displayed as a list on the user screen.

MessageSearch.java

This activity performs the functionality of searching through messages. When user gives input in the search box, it searches for the messages that contain the user string and the results are displayed as a list on the user screen.

AppSearch.java

This activity performs the functionality of searching through applications. When user gives input in the search box, it searches for the application that contains the user string and the results are displayed as a list on the user screen.

FileSearch.java

This activity performs the functionality of searching pdf, doc and ppt files. Only these three formats file are searched because the other format files can only be viewed with appropriate software or application installed on the device. When user gives input in the search box, it searches for the files in external memory that contain the user string in its title and the results are displayed as a list on the user screen.
SearchActivity.java

This activity performs the combined functionality of MusicSearch, MessageSearch, ContactSearch, AppSearch, FileSearch. When user gives input in the search box, it searches for the Music, Messages, Contacts, Applications and Files that contain the user string and displays the results as a list on the user screen.

AllInOneActivity.java

This activity performs the functionality associated with Bluetooth, Wi-Fi, Calculator, Screen Lock and Torch.

2.2.3 Classes

Result.java

This class defines the structure of each result entities that are returned by the search operations. It contains _ID, TITLE, LOCATION, TYPE as attributes of the resultant entities and some methods that are operated on these attributes.

ApplicationVariables.java

This class contains the list attributes as variables that store the list of results that are returned by the search operation.

ResultsListAdapter.java

This class performs the functionality that takes the user to original entity location with single touch of an entity in the result list.
3. IMPLEMENTATION OF MODULES

This project mainly consists of two modules and they are described below

- Search Module
- Collection Module

3.1 Search Module

It consists of search components through which a user can search for Contacts, Music, Messages, Applications and User files. In regular Android devices, we need to go to contacts to search for a contact, to media to search for a music file, to messages to search for a particular message etc. This module eliminates such user effort to move to a specific location for searching. It provides a global search through which user can search for Contacts, Music, Messages, Applications and User files. It searches for the given name in various locations and provides results of the search to the user. It also provides individual search operations for each of them. As this module consists of all the search components, it is named as Search Module.

This is the main module of this application and the search locations can be internal memory of the smartphone or external memory (SD Card). The main criteria for this search functionality is:

- Contacts
- Music
- Messages
- Applications
- User files
In Android smart phones data is stored in SQLite3 db files of format “.db”. The above-mentioned items are stored in specific locations either in internal memory or on external memory or SD card. The meta-data in data sources can be accessed using Content Providers. A content provider allows application to access data in SQLite or db file [18]. The implementation of the search modules is described below.

3.1.1 Contact Search

This component searches for the contacts that contain the user string in its name and displays the results as a list on the user screen. The contact information such as _ID, DISPLAY_NAME, HAS_PHONE_NUMBER is stored in ContactsContract.Contacts table. If a contact has one or more contact numbers then HAS_PHONE_NUMBER stores 1 else it stores 0. Then contact numbers associated with _ID are stored in ContactsContract.CommonDataKinds.Phone table [2]. The required information is retrieved into a cursor by querying the table as shown below.

```java
Cursor cursor = getContentResolver().query(ContactsContract.Contacts.CONTENT_URI, null, null, null, null);
```

The cursor contains all the information of the contact, but the information is filtered to retrieve _ID, DISPLAY_NAME, and PHONENUMBER. This retrieved information is stored into a list called resultsListContacts. If a contact has more than one contact number, all the numbers are retrieved, concatenated and stored them in resultsListContacts separating them by comma. The code snippet shown in figure 6 performs the mentioned process.

```java
Cursor cursor = getContentResolver().query(ContactsContract.Contacts.CONTENT_URI, null, null, null, null);
while (cursor.moveToNext()) {
    try{
        String image_uri = cursor.getString(cursor.getColumnIndex(ContactsContract.CommonDataKinds.Phone.PHOTO_URI))
    }
```
String contactId =
    cursor.getString(cursor.getColumnIndex(ContactsContract.Contacts._ID));
String name = cursor.getString(cursor.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME));
String hasPhone = cursor.getString(cursor.getColumnIndex(ContactsContract.Contacts.HAS_PHONE_NUMBER));
if (Integer.parseInt(cursor.getString(cursor.getColumnIndex(ContactsContract.Contacts.HAS_PHONE_NUMBER))) > 0) {
    Cursor phones = getContentResolver().query(
        ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
        null, ContactsContract.CommonDataKinds.Phone.CONTACT_ID + " = " + contactId,
        null, ContactsContract.CommonDataKinds.Phone.CONTACT_ID + " - 
        StringType, null, null);
    String num = "";
    while (phones.moveToNext()) {
        String phoneNumber = phones.getString(phones.getColumnIndex(
            ContactsContract.CommonDataKinds.Phone.NUMBER));
        num = num + phoneNumber + ",";
    }
}

Figure 6: Code snippet of Contact Search

The user string is matched with the display names of the contacts and if the
display name contains the user string then the contact information is stored in
sortedArray list as shown in figure 7. Once the result list is obtained it is populated on
to the user screen as a scroll list using populateListView() function.

textlength = searchBox.getText().length();
sortedArray.clear();
if (textlength != 0){
    for (int i = 0; i < lvarrModified.length; i++) {
        if (textlength <= lvarrModified[i].length() )
            //2 -- Any word -- even to a letter
            if (lvarrModified[i].toLowerCase().contains(searchBox.getText().toString().toLowerCase()))
                {
                    sortedArray.add(lvarrModified[i]);
                }

Figure 7: Code snippet of Contact Search

With a single touch of a contact in result list, user can view all the details of the
contact.

3.1.2 Music Search

This component searches for the music files that contain the user string in its title
and displays the results as a list on to the user screen. The information regarding music
files is stored in MediaStore.Audio.Media table [3]. There can be different types of music files such as ringtone, alarm, songs etc. So we can identify music or song files by an attribute called is_music, value 1 indicates music file and 0 indicates not [3]. First all the details of the music files in the device is retrieved into a cursor and stored in a list called resultsListMusic as shown in figure 8.

```
private void getAllMusicFiles() {
    String selection = MediaStore.Audio.Media.IS_MUSIC + " != 0"
    String[] projection = {
        MediaStore.Audio.Media._ID,
        MediaStore.Audio.Media.TITLE,
        MediaStore.Audio.Media.DATA
    }
    Cursor cursor = this.managedQuery(
        MediaStore.Audio.Media.EXTERNAL_CONTENT_URI,
        Projection,
        selection, null, null);
    while (cursor.moveToNext()) {
        ApplicationVariables.resultsListMusics.add(new Result(cursor.getString(0),
            cursor.getString(1), cursor.getString(2), "Music"));
    }
}
```

Figure 8: Code snippet of Music Search

The user string is matched with the titles of all music files. From all the files that contain user string in its title are stored in a temporary list called as sortedArray as shown in figure 9.

```
textlength = searchBox.getText().length();
sortedArray.clear();
if (textlength==0){
    for (int i = 0; i < lvarrModified.length; i++) {
        if (textlength <= lvarrModified[i].length()) {
            //2 -- any word -- even to a letter
            if (lvarrModified[i].toLowerCase().contains(searchBox.getText().toString().toLowerCase())){
                sortedArray.add(lvarrModified[i]);
            }
        }
    }
```

Figure 9: Code snippet of Music Search

Once the resultant list is obtained, the results are populated on the user screen using populateListView() function. With single touch of a specific music file
displayed in the list, user can play the respective music file. User can select only one file at an instance.

3.1.3 Message Search

This component searches for the messages that contain the user string in its body and displays the results as a list on to the user screen. All the messages are stored in “content://sms/inbox” location of the device. As there is no default content provider, we create an uri [6], which acts as a reference for the location. The details of all the messages such as THREAD_ID, PERSON, NUMBER, BODY of message is stored in a cursor using the uri [17]. Those details are filtered and required information such as THREAD_ID, PERSON, NUMBER, BODY are stored in the list called resultListMessages. The code snippet shown in figure 10 performs the mentioned process.

```java
Uri mSmsinboxQueryUri = Uri.parse("content://sms/inbox");
Cursor cursor1 = getContentResolver().query(mSmsinboxQueryUri,
    new String[] { "_id", "thread_id", "address", "person", "date", "body", "type" }, null, null, null);
String name = cursor1.getString(cursor1.getColumnIndex(columns[1]));
String date = cursor1.getString(cursor1.getColumnIndex(columns[2]));
String msg = cursor1.getString(cursor1.getColumnIndex(columns[3])); //msg data
String type = cursor1.getString(cursor1.getColumnIndex(columns[4]));
String threadid = cursor1.getString(cursor1.getColumnIndex(columns[5]));
ApplicationVariables.resultsListMessages.add(new Result(threadid, msg, "content://mms-sms/conversations/"+threadid, "SMSMessage"));
```

Figure 10: Code snippet of Message Search

All the message bodies that contain the user string are taken from resultListMessages and stored in a temporary list called SortedArray. The matched messages are populated as a list on to the user screen. The user can touch one of the messages in the list to go to specific message thread, which is done, using the THREAD_ID.
3.1.4 Application Search

This component searches for the installed applications in the device that contains the user string in its name and displays the results as a list on to the user screen. Each application is enclosed in a package on the Android device. A class called PackageInfo contains overall information about the packages [4]. At first all the information regarding the installed packages is retrieved using getInstalledPackages() method and stored in resultsListApps list which contains applicationname, Packagename, version number [4]. As we need only the applications that contain the user string, the user string is matched with application names in resultsListApps list as shown in figure 11. All the matched application information is displayed as a list on the user screen. The user can launch the application by single touch of the items in the result list.

```java
List<ApplicationInfo> listPacks = getPackageManager().getInstalledApplications(PackageManager.GET_META_DATA);
String installedPackage, title;
Intent launchActivity;
for(ApplicationInfo pac : listPacks){
    installedPackage = pac.packageName;
    launchActivity = getPackageManager().getLaunchIntentForPackage(pac.packageName);
    if(launchActivity != null){
        title = pac.loadLabel(getPackageManager()).toString();
        Result result = new Result(pac.icon, title, pac.packageName, "Application");
        result.setIcon(pac.loadIcon(getPackageManager()));
        ApplicationVariables.resultsListApps.add(result);
    }
}
```

Figure 11: Code snippet of Application Search

3.1.5 File Search

This component searches for the pdf, doc and ppt files in the external memory that contain the user string in its title and displays the results as a list on to the user screen. Only three formats pdf, doc and ppt are chosen in this search because only these format files can be viewed in smartphone device. The information regarding the files in the device is stored in MediaStore.Files.FileColumns table [5]. Files are
differentiated by MIME type. MIME (Multipurpose Internet Mail Extensions) is an Internet standard that is used to describe the contents of various files [18]. By using MIME\_TYPE attribute in the table, the details of the files that are in external memory are retrieved into a cursor and then stored in a list called resultsListFiles as shown in figure 12.

```java
ContentResolver cr = this.getContentResolver();
Uri uri = MediaStore.Files.getContentUri("external");
String[] projection = { "_id", "_data", "title", "mime_type"};
String mimeType = MimeTypeMap.getSingleton().getMimeTypeFromExtension("pdf");
String mimeType2 = MimeTypeMap.getSingleton().getMimeTypeFromExtension("doc");
String mimeType3 = MimeTypeMap.getSingleton().getMimeTypeFromExtension("ppt");
String[] selectionArgsPdf = new String[]{mimeType, mimeType2, mimeType3};
Cursor allPdfFiles = cr.query(uri, projection, selectionMimeType, selectionArgsPdf, null);
allPdfFiles.moveToFirst();
while (allPdfFiles.moveToNext()) {
    ApplicationVariables.resultsListFiles.add(new Result(allPdfFiles.getString(0),
        allPdfFiles.getString(2), allPdfFiles.getString(1), "Document"));
}
```

Figure 12: Code snippet of File Search

Once the information of all the files is obtained in resultsListFiles, the user string is matched with the titles of the files. The files that contain the user string in its title are stored in SortedArray list. These files are then populated as a list on to the user screen. With single touch of a file in the list user can view it on screen.

### 3.1.6 Global Search

This component performs the combined functionality of ContactSearch, MusicSearch, MessageSearch, ApplicationSearch and FileSearch. When user gives input in the search box, it searches for the Contacts, Music, Messages, Applications and Files that contain the user string and displays the results as a list on to the user screen. This search combines the results from each search and displays them as a single list on the user screen.
3.2 Collection Module

This module consists of basic smart phone functionalities that are mostly used by Android phone users. The functionalities that are implemented as a part of this module are:

- Bluetooth
- Wi-Fi
- Calculator
- Screen Lock
- Flash Light

The user can turn on/off their Bluetooth, Wi-Fi using this module. The Calculator will be readily available to perform arithmetic operations. The user can Lock the screen by using screen lock and turn on/off the mobiles LED light to use it as a torch by clicking the Flashlight button. In this way some basic mobile functionalities are provided in a single application for faster and better usage. The detailed description of these functionalities is provided in the following session.

3.2.1 Bluetooth

Bluetooth service helps mobile devices to connect to other devices. This application uses BluetoothAdapter class that performs fundamental Bluetooth tasks [8]. The following permissions are required in order to access the devices Bluetooth features.

```xml
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
```

An object of BluetoothAdaptor class is created and if it is null then the device doesn’t support Bluetooth functionality. If the device supports Bluetooth then it can be
enabled using `enable()` function and disabled using `disable()` function [8]. The Code snippet shown in figure 13 performs this task.

```java
if(!mBluetoothAdapter.isEnabled()){  
  mBluetoothAdapter.enable();  
  Toast.makeText(this, "BlueTooth ON!", Toast.LENGTH_LONG).show();  
  bluetoothTextView.setText("Switch OFF Bluetooth");  
} else {  
  if (mBluetoothAdapter.isEnabled()) {  
    mBluetoothAdapter.disable();  
    Toast.makeText(this, "BlueTooth OFF!", Toast.LENGTH_LONG).show();  
    bluetoothTextView.setText("Switch ON Bluetooth");  
  }  
}
```

**Figure 13: Code snippet of Bluetooth**

### 3.2.2 Wi-Fi

Wi-Fi enables mobile devices to be connected to Internet or Wi-Fi resources [9]. The following permissions are required in order to access Wi-Fi features of the device.

```xml
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
```

The implementation of Wi-Fi is achieved using toggle button and involves using `WifiManager` class that manages all activities of Wi-Fi connectivity. It can be enabled and disabled using `setWifiEnabled()` function and takes Boolean variable as input [9]. If the input is true then Wi-Fi is turned on and if the input is false then it is turned off as shown in figure 14.

```java
WifiManager wifiManager = (WifiManager) this.getSystemService(Context.WIFI_SERVICE);  
if (status == true && !wifiManager.isWifiEnabled()) {  
  wifiManager.setWifiEnabled(true);  
} else if (status == false && wifiManager.isWifiEnabled()) {  
  wifiManager.setWifiEnabled(false);  
}
```

**Figure 14: Code snippet of Wi-Fi**

### 3.2.3 Calculator

This is a basic application that is used to perform arithmetic calculations. Many users in their day-to-day life use this application for arithmetic calculations. Mostly all
android mobiles consist of a calculator application in it. This component makes use of the inbuilt calculator app in the device. When user touches the calculator button on the screen it takes the user to the default calculator application on the device to perform arithmetic calculations. The built-in calculator app contains a package on the device. It is retrieved using package manager [10] and the package is invoked using an intent [7] to launch the calculator app. If there is no such package then the device doesn’t consist of the calculator app and so the application throws an error message to user. The code snippet shown in the figure 15 performs the mentioned task.

```java
if(items.size()>=1){
    String packageName = (String) items.get(0).get("packageName");
    Intent i = pm.getLaunchIntentForPackage(packageName);
    if (i != null)
        startActivity(i);
    else{
        Toast.makeText(this, "No app found", Toast.LENGTH_LONG).show();
    }
}
```

Figure 15: Code snippet of Calculator

3.2.4 Screen Lock

This functionality locks or inactivates the display screen. As the smartphones are touch screens, in order to eliminate unwanted usage of applications due to unintentional touches, the user can lock or disable the screen. The following permissions are required for accessing screen lock features [11].

```xml
<uses-permission android:name="android.permission.WAKE_LOCK" />
```

The implementation of this application involves using DevicePolicyManager class, which contains Locknow() function [11]. The device manager needs to be enabled before calling Locknow() function as shown in figure 16. Then Locknow() function is called to lock the device which indicates that the lock screens timeout value expired.
deviceManager = (DevicePolicyManager) getSystemService(Context.DEVICE_POLICY_SERVICE);
activityManager = (ActivityManager) getSystemService(Context.ACTIVITY_SERVICE);
compName = new ComponentName(this, MyAdmin.class);
boolean active = deviceManager.isAdminActive(compName);
if (active){
deviceManager.lockNow();
}

Figure 16: Code snippet of Screen Lock

3.2.5 Flashlight

This functionality is achieved by using the camera LED light as Flashlight. Implementation of this application uses Camera class to access hardware LED light of the mobile. The following permissions are required to access the device’s camera hardware.

<uses-permission android:name="android.permission.CAMERA" />
<uses-feature android:name="android.hardware.camera" />

At first it is checked whether the device supports camera using isCameraSupported() function which returns a Boolean. If it returns false then camera is not supported and error message is shown to user. This application turns on the LED light by setting \texttt{FLASH\_MODE\_TORCH} parameter of the camera class \cite{12}. Setting \texttt{FLASH\_MODE\_OFF} parameter of this class can turn it off as shown in figure 17. The implementation of flashlight is done using a special button called Toggle button which it self consists of a light to indicate on/off.

```java
if (on) {
    Log.i("info", "torch is turn on!");
p.setFlashMode(Parameters.FLASH_MODE_TORCH);
camera.setParameters(p);
camera.startPreview();
} else {
    Log.i("info", "torch is turn off!");
p.setFlashMode(Parameters.FLASH_MODE_OFF);
camera.setParameters(p);
camera.stopPreview();
}
```

Figure 17: Code snippet of Flashlight
4. TESTING AND EVALUATION

In this phase the functionality of the developed application is tested and evaluated. The testing of the application is performed on LG Nexus 4, which is an Android device. Each of the components in the application is tested and the results are provided.

4.1 Testing of Search Modules

4.1.1 Contact Search

The Contact search functionality searches for the contacts in mobile device. It searches according to the input string given by the user. If the display name of the contact contains the input string then it is displayed. If the matched contacts contain more than one contact number then all of numbers are displayed separated by comma. All the results are displayed as a list. With single touch of specific contact in result list, user can view complete details of the contact as shown in figure 18.

![Testing of Contact Search](image)

Figure 18: Testing of Contact Search
4.1.2 Music Search

The Music search functionality searches for the music in the device. If the title of the music file contains the user string then it is displayed in the result list. Single touch of a music file in the result list allows user to play it. In Figure 19 user string is given as “su” and results are listed as shown. One of the music files in the list is played by a single touch of the file.

Figure 19: Testing of Music Search
4.1.3 Message Search

This functionality allows the user to search for the messages in the device. If the body of the message contains the user string then it is displayed in the result list. Single touch of a message in result list allows user to go to the respective thread of the message. In figure 20, user string “see” is given as input and messages that contain user string in its body are displayed as list.

Figure 20: Testing of Message Search
4.1.4 Application Search

This component allows the user to search for the applications that are installed on the device. If the name of the application contains the user string then it is displayed in the results list. User can launch an application by single touch of an item in result list as shown in the figure 22.

![Application Search](image)

Figure 22: Testing of Application Search
4.1.5 File Search

This Functionality allows the user to searches for pdf, doc and ppt files on the device. If the title of the file contains the user string then the file is displayed in the results list. Figure 21 shows the matched file when the user string “m” is given as input.

![Figure 21: Testing of File Search](image)

Figure 21: Testing of File Search
4.1.6 Global Search

This search is combination of Contacts, Music, Messages, Applications and Files. User can search for all these things using this search. The entities that contain the user string are displayed in the result list. As the result contains different types of entities, their type such as music, contact, SMS, apps or document is displayed as shown in figure 23.

Figure 23: Testing of Global Search
4.2 Testing of Collection Module

Figure 18 shows the home screen of the application that contains Search modules and Collection module. Once the Collection module button is clicked it takes user to the screen, which contains Bluetooth, Wi-Fi, Calculator, Lock Screen and Flashlight functionalities as shown in figure 24.

![Home Screen of Application](image)

Figure 24: Testing of Collection Module
4.2.1 Bluetooth

The Bluetooth button in collection module is used to turn Bluetooth on or off. It is also implemented using the toggle button. After the Bluetooth is turned on/off a dialog box is displayed indicating the status of the Bluetooth as shown in figure 25.

Figure 25: Testing of Bluetooth
4.2.2 Wi-Fi

The Wi-Fi button in the collection module is used to turn Wi-Fi on or off. A special button called Toggle button is used which indicates the on/off status of the button by a small green light on it. After the Wi-Fi is turned on/off a dialog box is displayed indicating the status of the Wi-Fi as shown in figure 26.

Figure 26: Testing of Wi-Fi
4.2.3 Calculator

The Calculator button in collection module opens the inbuilt calculator app on the device. Once the user touches the calculator button, it takes user to the calculator screen and the user can perform calculations as shown in figure 27.

Figure 27: Testing of Calculator
4.2.4 Flash Light

The Flashlight button in collection module is used to turn Flash on or off. The figure 28 shows the turned on flashlight.

![Figure 28: Testing of Flash Light](image)

4.2.5 Screen Lock

Screen Lock button in collection module disables the mobile device screen. As smart phones are touch accessible, it prevents unwanted access.
5. CONCLUSION AND FUTURE WORK

This project presents the implementation of an android application that brings the most frequently used functionalities on smartphone to a single app. It contains two main modules, one is Search and the other is collection module. The Search module provides a wide range of entity search such as Contacts, Music, Messages, Applications and User files on the device. The collection module provides access to different smartphone functionalities such as Bluetooth, Wi-Fi, Calculator, Screen lock and Flashlight. No additional memory is required for this application, as it does not stores any information. In this way this application provides faster and better way to access almost all the entities on the device from a single application saving users time and effort.

The future work of this application involves adding some more smartphone functionalities to it. The search can be improvised such that it searches for images, videos, messages in Facebook, WhatsApp and other messaging clients. The Email cannot be implemented because Google has blocked third party applications access to Gmail app on the device for security reasons. There are no Android API’s that can access other email clients to implement Email Search functionality. It can only be done by implementing our own email client and storing the emails in a database and accessing the database by this application, but it is entirely another project. The Email Search can be implemented in future if Google provides Android API’s to access email clients on the device. The user interface of the application can be improved in an attractive way and user friendly by using more Android Widgets.
BIBLIOGRAPHY AND REFERENCES


Appendix A

A.1 Performance Testing on Android Devices

Scaling of performance is done by three variables Good, Moderate, Slow. The feedback of the user is given in the below table.

User 1 - Samsung Galaxy Note
User 2 – LG Nexus 4
User 3 – LG G2
User 4 – LG Nexus 5
User 5 – Samsung Galaxy S III

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>User 1</td>
</tr>
<tr>
<td>1) Contact Search</td>
<td>Good</td>
</tr>
<tr>
<td>2) Music Search</td>
<td>Good</td>
</tr>
<tr>
<td>3) Message Search</td>
<td>Good</td>
</tr>
<tr>
<td>4) Application Search</td>
<td>Good</td>
</tr>
<tr>
<td>5) File Search</td>
<td>Good</td>
</tr>
<tr>
<td>6) Global Search</td>
<td>Moderate</td>
</tr>
<tr>
<td>7) Wi-Fi</td>
<td>Good</td>
</tr>
<tr>
<td>8) Bluetooth</td>
<td>Good</td>
</tr>
<tr>
<td>9) Calculator</td>
<td>Good</td>
</tr>
<tr>
<td>10) Screen Lock</td>
<td>Good</td>
</tr>
<tr>
<td>11) Torch</td>
<td>Good</td>
</tr>
</tbody>
</table>