AUTO PROFILE

by

SRIKAR ANUMULA

B.Tech., Sri Indu College of Engg & Tech, 2013

A REPORT

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computing and Information Sciences
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2015

Approved by:

Major Professor
Daniel A. Andresen

Abstract

Present times most of the applications allow users to set profiles and activate them manually. Also, there is no such application to alert the calling person automatically whenever we are busy in a meeting. Auto Profile is an application which addresses these issues. Using this application, a custom profile can be created and activated. The user of this application can create any number of custom profiles with their own timings depending on their requirements from the available set of ringer modes such as vibration, silent, normal modes. All these settings will be changed automatically when they define a profile and set it as active.

Every profile can be scheduled according to their wish. Also, users can define how often and how long they would like to activate the selected profile. Auto Profile Switcher will recognize their profile based on the current time and it will apply the right profile at the right time according to the given settings. For instance, they can save a home profile and a work profile, and the application will automatically switch from one to other automatically, without any user action. It's also possible for the users to apply manually saved profile wherever they are.

Also, the application allows users not to bother about the calls from important persons when they are in some important meeting or busy. Automatically profile is activated as scheduled and sends a respective alert to the caller according to the profile activated. Auto Profile application is user-friendly and easy to use.

The proposed application is developed for the android platform, which is used to create custom profiles and make sure that the tasks are executed as scheduled. This relieves users as they do not need to bother about the calls when they are in some important meeting or busy.

As this is a mobile application users can easily organize their action wherever they are with ease.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my heart-felt gratitude to my **Parents** without whom I would not have been privileged to achieve and fulfill my dreams.

I am grateful to my Major Professor, **Dr. Daniel A. Andresen** who had the major hand in enabling me to do this project. I profoundly thank **Dr. Doina Caragea**, and **Dr. Torben Amtoft** for their help and for serving on my committee.

I would like to express my gratitude to all the people behind the screen who helped me to transform an idea into a real application. Last but not least I would like to acknowledge the academic support received from the staff and students of Kansas State University.

Chapter 1 - Introduction

Using this application a custom profile can be created with our own timings, that is depending on user’s requirements by using a set of profile settings modes such as vibration, silent, normal. All these settings will be changed automatically when they define a profile and set it as active.

There are some predefined profiles: Silent, normal and vibrate, as well as unlimited number of user defined profiles based on timings. Every profile can be scheduled according to their wishes. So one can define how often and for how long would they like to activate the selected profile.

Auto Profile Switcher will recognize your profile based on given time and it will apply the right profile, according to the settings. For instance, users can save a home profile and a work profile, and the application will switch from one to other automatically, without any user’s action. It's also possible for the user to apply manually saved profile wherever you are. Auto Profiles application is user-friendly and easy to use.

1.1 Purpose

The application recognizes your profile based on your time and it will apply the right profile, according to given settings. For case in point, a user can create and save a custom profile like a work profile, Home Profile so the application will switch from one to other automatically, without any user’s action. The application allows the user to apply manually saved profile wherever they are. The application is built using Android SDK, which makes the application run when deployed on any android powered mobile phones.

1.2 Scope

One of the fastest growing industries now a day are mobile industry. There are many competitors in their area who are doing research and development on new platforms & user experience. One such technology is Android from Google. These phones are described as next Generation mobiles. Developing application for such mobile phones using the open source android SDK with a well-known concept Getting Things done is quite interesting. This application makes one to concentrate on performing their tasks instead of concentrating on their mobile phones.

1.3 Project Overview

The application provides a service to the end user to make a custom profile and activate it automatically. An auto profile process contains scheduler which will automatically shift the profile with respective to time and, the application will switch from one to other automatically, without any user action. It's also possible for the user to apply manually saved profile wherever they are. If they have not attended the call, then the application will automatically send the respective message to concerned calling person depending on the profile set. Auto Profile application is user-friendly and easy to use.

Chapter 2 - Background & System Analysis

2.1 Android OS & SQLite

The android operating system is basically an operating system for mobiles. It is mobile operating system that uses a modified version of the Linux kernel 2.6. Google developed Android as part of the Open Handset Alliance, a group of more than 30 mobile and technology companies working to open up the mobile handset environment.

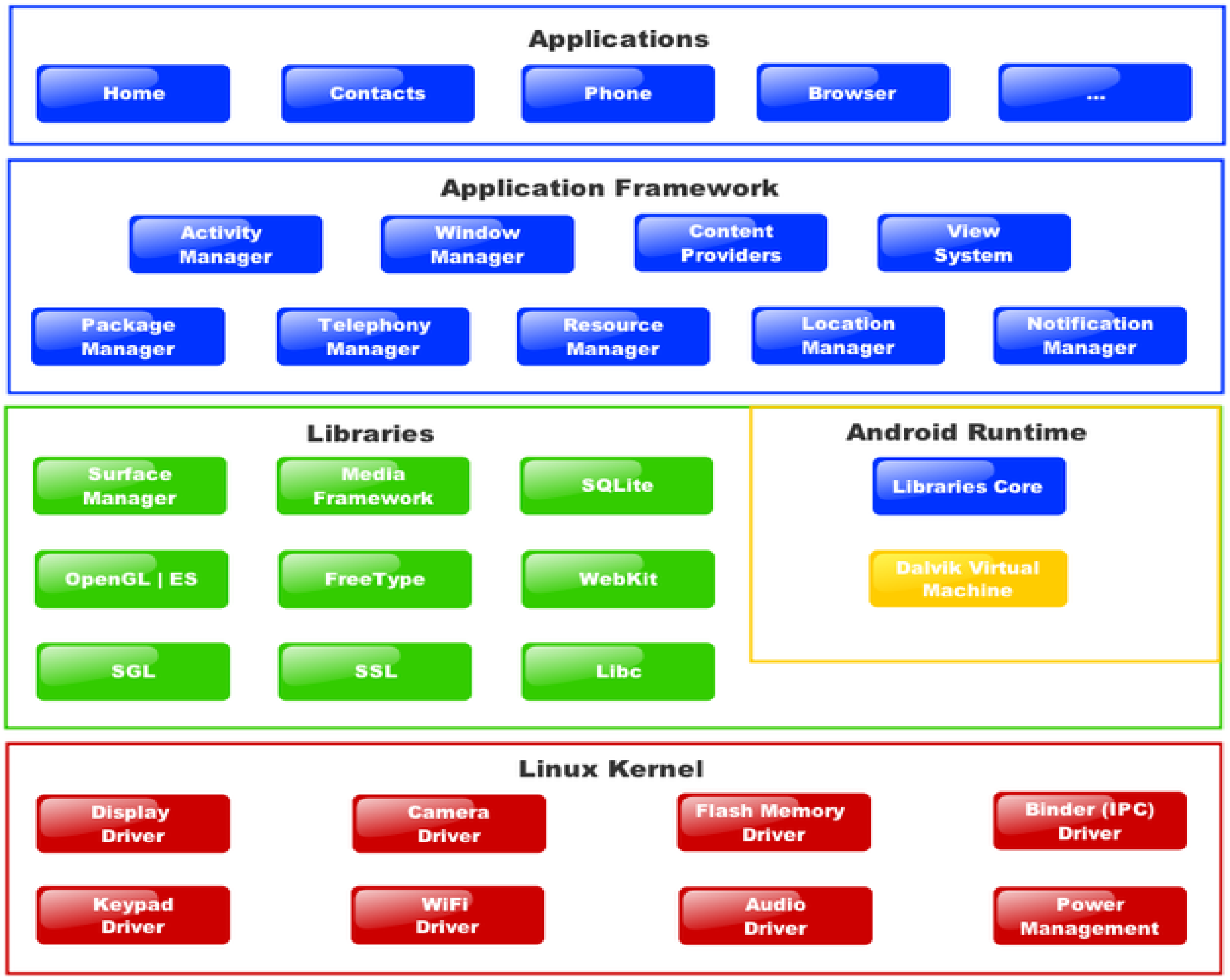


Figure 1 Android Architecture [2]

It is an open source software platform and operating system for mobile devices. Based on the Linux kernel. Developed by Google and later the Open Handset Alliance (OHA). Allows writing managed code in the Java language. Android has its own virtual machine i.e. DVM (Dalvik Virtual Machine), which is used for executing the android application. Android is a free downloadable open source software stack for mobile devices that include an Operating system. Android OS is developed under a code name based on dessert items.

The software stack is split into four layers called the application layer, the application framework, the libraries and runtime and the kernel.

Figure 1 above depicts the different layers and component of the Android operating system. Each layer offers varied services to the layer above it. The different layers in the architecture are:

2.1.1 Linux Kernel

The architecture is based on the Linux2.6 kernel. This layer is the core of the android architecture. It provides service to power management, memory management, security etc. It helps in software or hardware binding for better communication.

2.1.2 Native Libraries

Android has its own libraries, which is written in C/C++. These libraries cannot be accessed directly. With the help of application framework, we can access these libraries. There are many libraries like web libraries to access web browsers, libraries for android and video formats etc.

2.1.3 Android Run Time

Dalvik virtual machine

The Android Runtime was designed specifically for Android to meet the needs of running in an embedded environment where you have a limited battery, limited memory, limited CPU. Dalvik is the process virtual machine in Google's android operating system. It is the software that runs the apps on android devices. Dalvik is thus an integral part of android, which is typically used on mobile devices such as mobile phones and tablet computers. Programs are commonly written in java and compiled to bytecode.

Core libraries

This is in blue, meaning that it's written in the Java programming language. • The core library contains all of the collection classes, utilities, IO, all the utilities and tools that you’ve come to expected to use.

2.1.4 Application Framework

This is all written in a Java programming language and the application framework is the toolkit that all applications use. These applications include the ones that come with a phone like the home applications or the phone application. It includes applications written by Google, and it includes apps that will be written by you. So, all apps use the same framework and the same APIs.

Activity manager

It manages the lifecycle of applications. It enables proper management of all the activities. All the activities are controlled by activity manager.

Resource manager

It provides access to non-code resources such as graphics etc.

Notification manager

It enables all applications to display custom alerts in a status bar.

Location manager

It fires alerts when a user enters or leaves a specified geographical location.

Package manager

It is used to retrieve the data about installed packages on the device.

Window manager

It is used to create views and layouts.

Telephony manager

It is used to handle settings of network connection and all information about services on the device.

2.1.5 Application Layer

The final layer on top is Applications. It includes the home application, the contacts application, the browser, and apps. It is the uppermost layer in the android architecture. All the applications like camera, Google maps, browser, SMS, calendars, contacts are native applications. These applications work with end-user with the help of application framework to operate.

2.2 SQLite

SQLite is an Open Source database. SQLite supports standard relational database features like SQL syntax, transactions, and prepared statements. The database requires limited memory at runtime (approx. 250 Kbyte) which makes it a good candidate from being embedded into other runtimes.

SQLite supports the data types Text (similar to String in Java), INTEGER (similar to long in Java) and REAL (similar to double in Java). All other types must be converted into one of these fields before getting saved in the database. SQLite itself does not validate if the types written to the columns are actually of the defined type, e.g. you can write an integer into a string column and vice versa.

2.2.1 SQLite in Android

SQLite is embedded into every Android device. Using the SQLite database in Android does not require a setup procedure or administration of the database.

You only have to define the SQL statements for creating and updating the database. Afterward, the database is automatically managed for you by the Android platform.

Access to the SQLite database involves accessing the file system. This can be slow. Therefore, it is recommended to perform database operations asynchronously.

2.3 System Analysis

2.3.1 Problem Definition

Compared to any normal applications this is that the application developed based on problems which we are facing in real time. This application was a small idea for the mobile device users. Every user had faced some difficulties when they receive calls when they are busy or in some important meeting.

The user has to remember the thing that needs to be done. Here in methodize the user can store the information on the mobile device so that they can get the things done according to the scheduled time.

Existing System

Generally user needs to perform the profile activation manually.

Proposed System

The proposed application is developed for the android platform, which is used to create a custom profile and organize tasks as what was scheduled. This relieves users as they need not bother about the calls when they are in some important meeting or busy. Then automatically profile is activated as what was schedule and sends a message to concerned call person. As this is a mobile application one can easily organize the action where ever the user is with ease.

Characteristics

The End user of the application is the mobile phone user. The user can organize their actions by creating their own custom profile with respective to time.

2.3.2 Requirements gathering

There are many applications online which help the user to activate the profiles automatically. But the main motto of this application is to allow the users to create the profiles as required and the application itself shift the profile based on the system time. Also, there are no such applications that notify the caller about the user with the help of a message. Many similar application online are checked thoroughly and their user interface is examined. All the basic requirements that are required for such similar applications are noted down. Also many suggestions from my friends are considered before starting my work on the application. The final overview of the application's requirements were explained to my major professor Dr. Daniel Andresen and then I started the development. The important requirements of the application are as follows.

- The user should be able to create any number of custom profiles based on his requirements.
- Any profile can be updated at any point of time by the user. All the validations are performed at this point of time.
- The user should be able to delete any specific or all the profiles at any point of time.

- The application should be able to start even after the mobile device is turned off and then turned on.
- The caller should be sent a message that the user has set for that specific profile time.
- The profiles should shift as scheduled by the user.

2.3.3 Requirement Specification

Software Requirements

There are different software requirements for developing and running the Android application. They are as follows.

For developing the application:

- Operating System: Windows 8 or higher
- Database: SQLite Database
- Tools: Eclipse SDK 3.8, ADT plug-in for eclipse
- Technologies used: Java, SQLite, Android
- Android Emulator: SDK Version 2.2 or higher

For running the application on Android device:

- Android smartphone with 2.2 or higher versions

Hardware Requirement

The distinct hardware requirements for developing and running an Android application are as follows:

For the development of the application:

- 256 MB RAM
- Minimum 250MB disk space

Hardware Requirements for running the application:

- Android Smart Phone 2.2 or higher
- 4.0MB space for execution

Chapter 3 - System Architecture and Design

3.1 System Architecture

The following is the system architecture of the Auto Profile application.

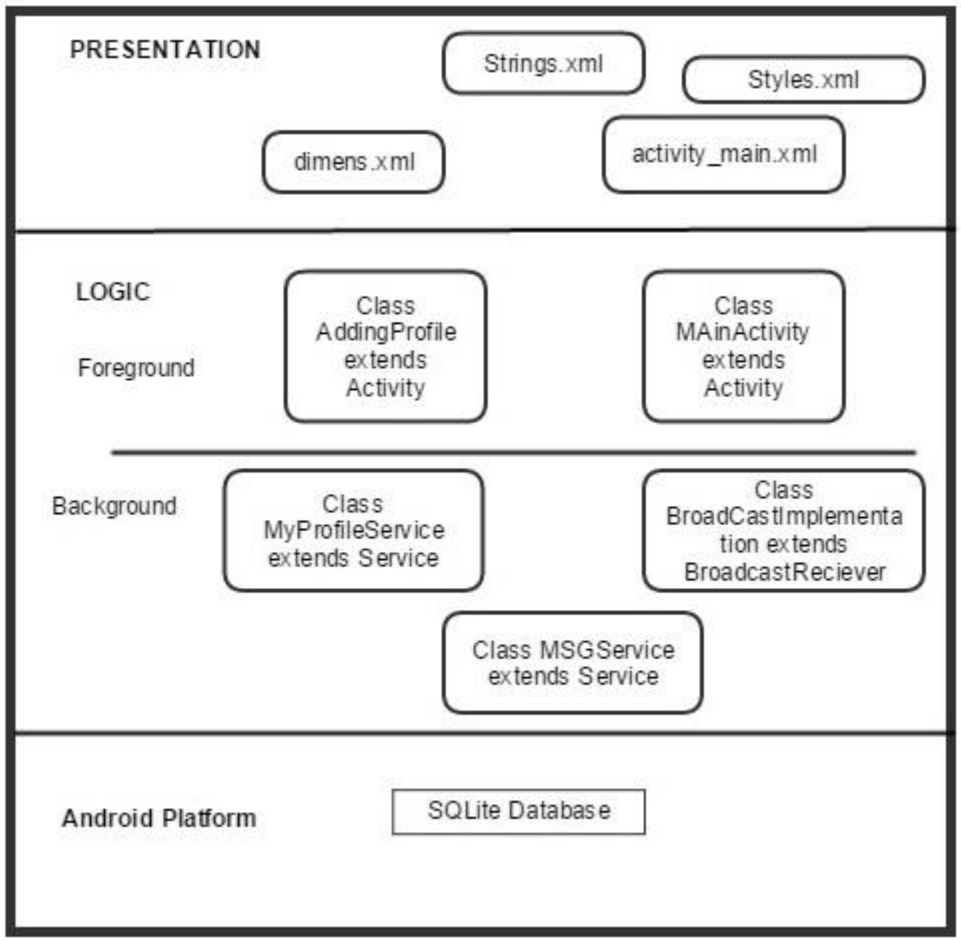

PRESENTATION
Strings.xml
Styles.xml
dimens.xml
activity_main.xml
LOGIC
Foreground
Class AddingProfile extends Activity
Class MAinActivity extends Activity
Background
Class MyProfileService extends Service
Class BroadCastImplementation extends BroadcastReciever
Class MSGService extends Service
Android Platform
SQLite Database

Figure 2 Architecture Diagram

The users can deploy the APK file and install the application. Once installed they can open it and create any number of custom profiles. Each profile will have a specific name and respective message. Once the profile is created it can be updated. The profile is scheduled for a specific time with a specific message. There are three available ring modes like silent, vibrate and normal. Also at any point of time user can delete a specific profile or all the profiles. The user can restart the service if the mobile is switched off by just clicking the activate button.

In the background, there is a class running which acts as a Broadcast receiver and another class which acts as a Service component which helps to send the message to the caller. The Broadcast receiver is used to get the phone state and sends the required information to the Service component with the help of Intent object. In the Service class, the respective message is retrieved from the database and the message is sent.

The profiles created using the user interface are stored in the SQLite database which will be created on the device's virtual memory using the java classes. The data is stored in the form of traditional tables and is retrieved using SQL commands.

3.2 System Design

3.2.1 Use Case Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases.

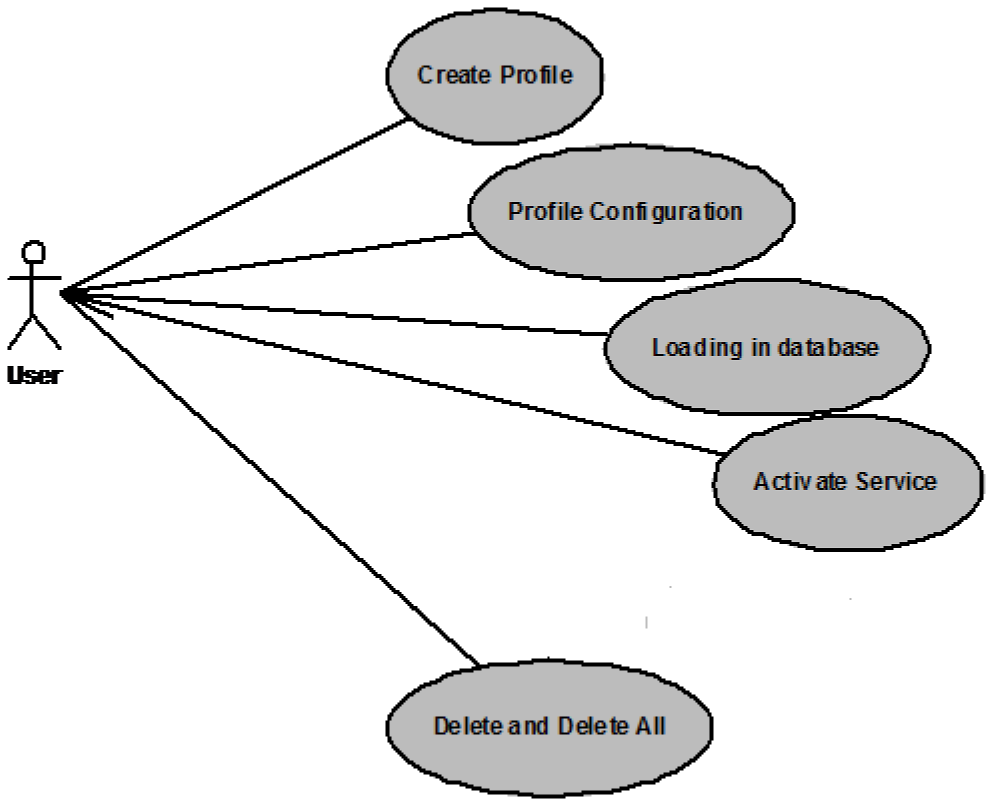


Figure 3 Use Case Diagram

In the above Use case diagram Actor represents the user that is using the system. He will be able to perform the different operations on the system and also the use cases represents the system in brief.

The Actor will be able to perform the operations like creating a custom profile, configuring profile, Loading Database and retrieving data, Activate Service and Delete a specific or all profiles.

3.2.2 Sequence Diagram

UML sequence diagrams are used to represent the flow of messages, events and actions between the objects or components of a system. Time is represented in the vertical direction showing the sequence of interactions of the header elements, which are displayed horizontally at the top of the diagram. Sequence diagrams are useful design tools because they provide a dynamic view of the system behavior which can be difficult to extract from static diagrams or specifications.

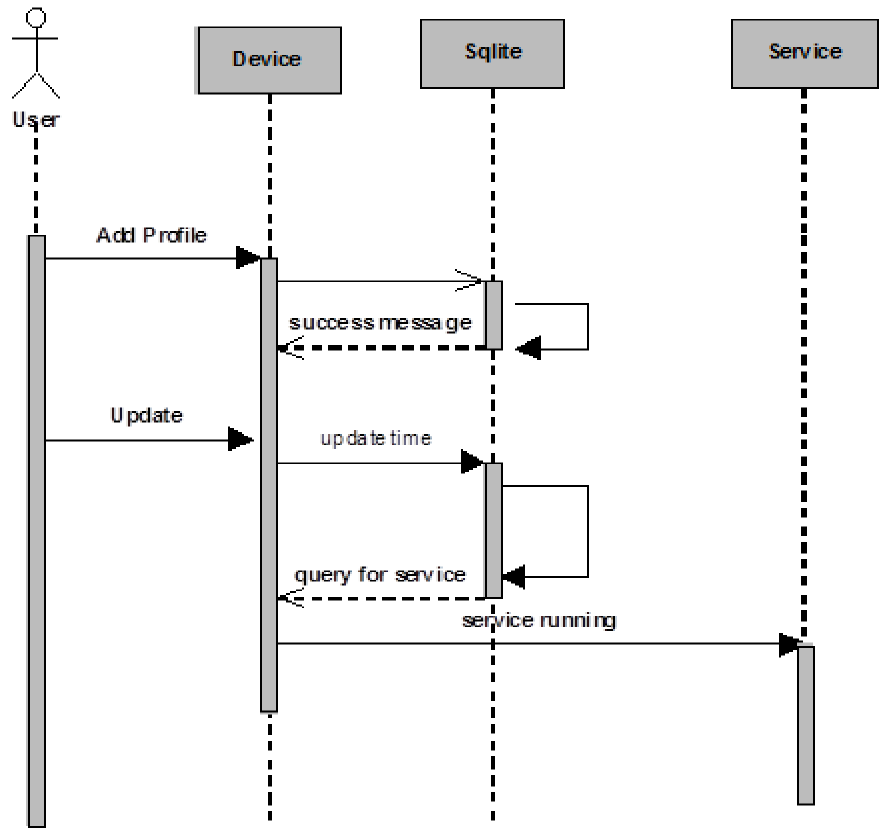


Figure 4 Sequence Diagram

3.2.3 Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

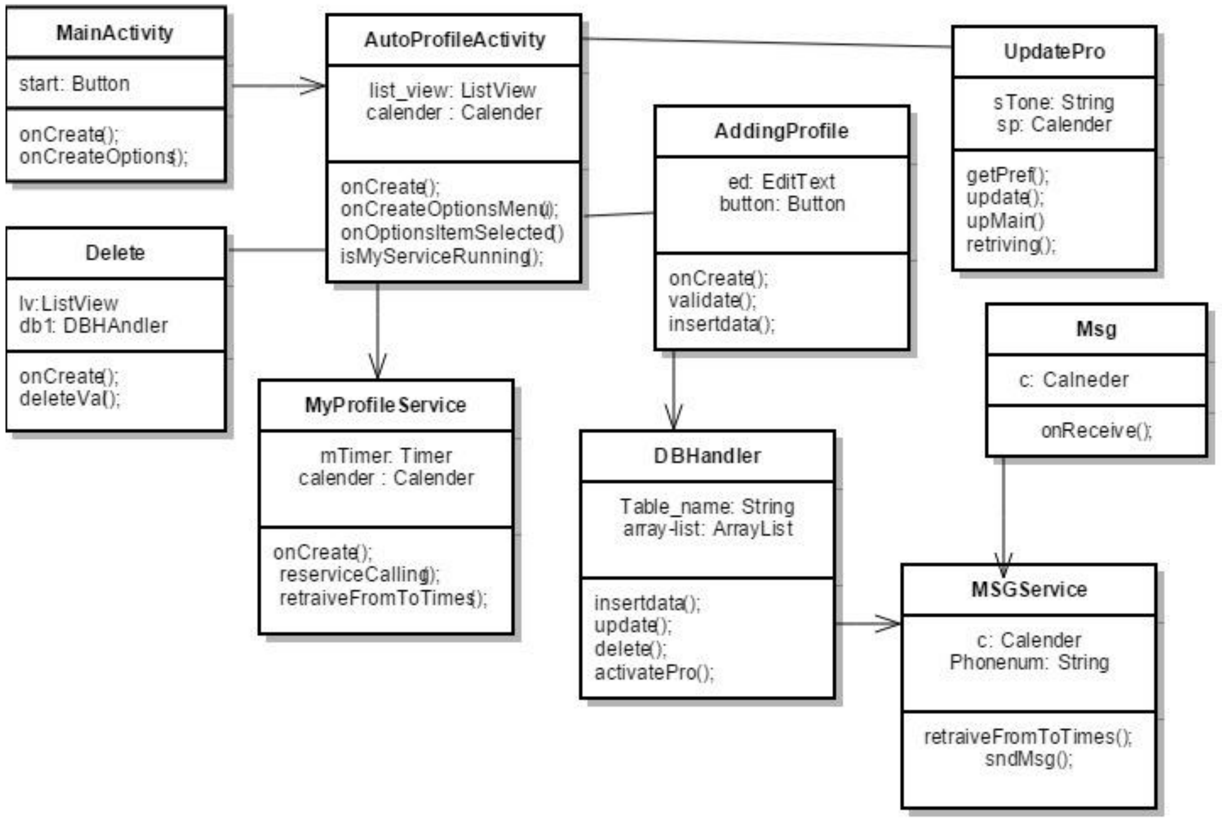


Figure 5 Class Diagram

Chapter 4 - Android framework Components

4.1 AndroidManifest.xml

Every application must have an AndroidManifest.xml [4] file in its root directory. The manifest file presents essential information about your app to the Android system, information the system must have before it can run any of the app's code.

The manifest does the following:

- It names the Java package for the application. The package name serves as a unique identifier for the application.
- It describes the components of the application — the activities, services, broadcast receivers, and other things that the application is composed off.
- It determines which processes will host application components.
- It declares which permissions the application must have in order to access protected parts of the API and interact with other applications.
- It also declares the permissions that others are required to have in order to interact with the application's components.
- It declares the minimum level of the Android API that the application requires.
- It lists the libraries that the application must be linked against.

The following is the AndroidManifest.xml file of the Auto Profile application.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.srikar.autoprofile"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="8"
        android:maxSdkVersion="18" />
    <uses-permission android:name="android.permission.CALL_PHONE" >
    </uses-permission>
    <uses-permission android:name="android.permission.READ_SMS" />
    <uses-permission android:name="android.permission.SEND_SMS" />
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name="com.srikar.autoprofile.AutoprofileActivity"
            android:label="@string/app_name" >
        </activity>
        <activity
            android:name="com.srikar.autoprofile.Info"
            android:label="@string/app_name"
            android:screenOrientation="portrait"
            android:theme="@style/CustomTheme" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
            </intent-filter>
        </activity>
        <activity
            android:name="com.srikar.autoprofile.AddingProfile"
            android:theme="@android:style/Theme.Dialog" >
        </activity>
        <service android:name="com.srikar.autoprofile.MyProfileService" >
        </service>
        <activity android:name="com.srikar.autoprofile.UpdatePro" >
        </activity>
        <service android:name="com.srikar.autoprofile.MyProfileService" >
        </service>
        <activity android:name="com.srikar.autoprofile.Delete" >
        </activity>
        <service android:name="com.srikar.autoprofile.MSGService" >
        </service>
        <receiver
            android:name="com.srikar.autoprofile.Broadcastimplimentation"
            android:process=":remote" >
        </receiver>
        <receiver android:name="com.srikar.autoprofile.Msg" >
            <intent-filter>
                <action android:name="android.intent.action.PHONE_STATE" />
            </intent-filter>
        </receiver>
        <activity
            android:name="com.srikar.autoprofile.MainActivity"
            android:label="@string/title_activity_main" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
      </manifest>
```

The manifest file has the name of the package and the application version details. It describes the minimum level of the Android API that is required and the maximum level of API that the application supports. The minimum SDK used for the application is 8 and the maximum

SDK used is 18. Next the permissions that the application is required to have are described. These permissions allow the application to interact with the other components of the mobile device. The permissions the application must have in order to access protected parts of the API and interact with other applications are Call phone permission, Read and Send SMS permissions and Read phone state permissions. The manifest file also defines the icon for the Auto profile application. Also, the manifest file defines different activities, services, and broadcast receivers. The intent-filter Main is the starting activity of the application.

4.2 Activities

An activity is a user interface concept. An activity usually represents a single screen in your application. It generally contains one or more views, but it doesn’t have to. An Activity is pretty much like it sounds—something that helps the user do one thing—and that one thing could be viewing data, creating data, or editing data. Most Android applications have several activities within them.

4.2.1 Life cycle of Activity

Android is sensitive to the lifecycle of an application and its components. Therefore, you’ll need to understand and handle lifecycle events of each activity in order to build a stable application. The processes running your Android application and its components go through various lifecycle events, and Android provides callbacks that you can implement to handle state changes.

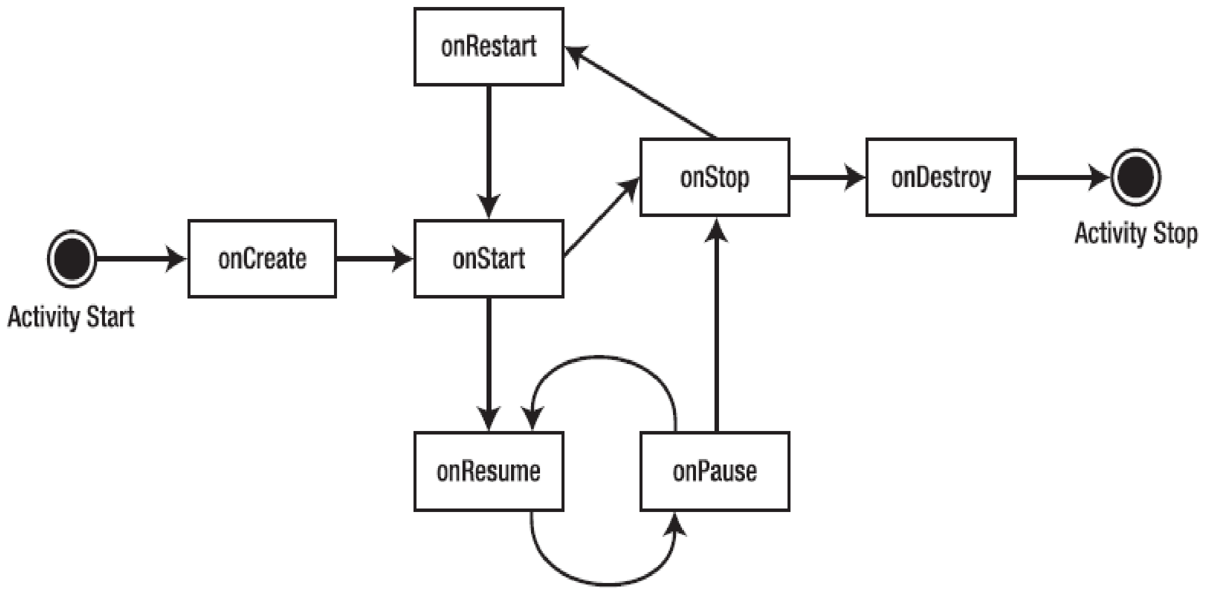
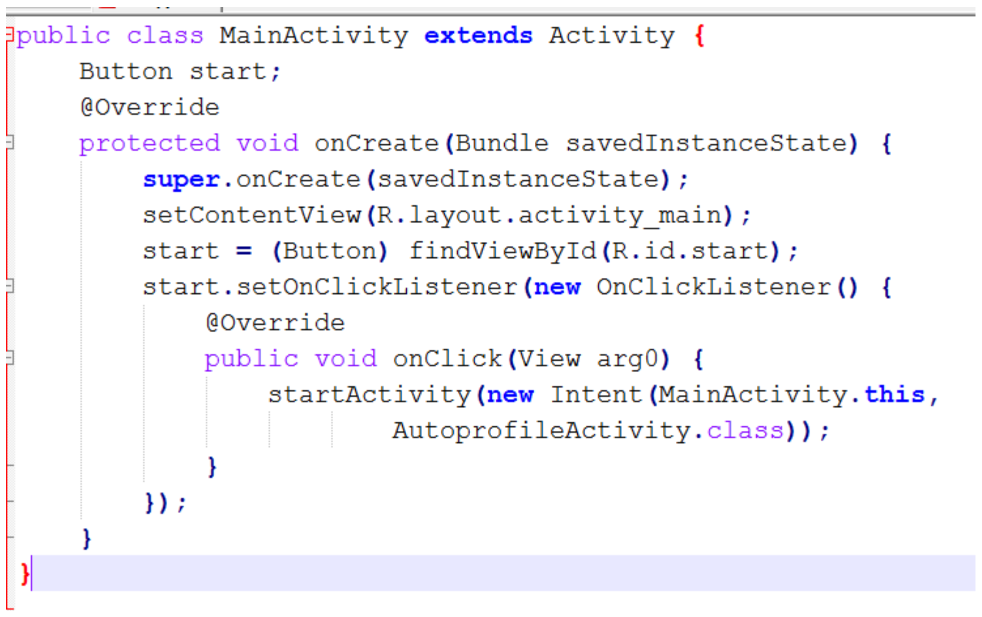


Figure 6 Activity Life Cycle

The various callback methods provided are shown in the rectangular boxes which are implemented to perform operations when an activity moves from one state to the other.

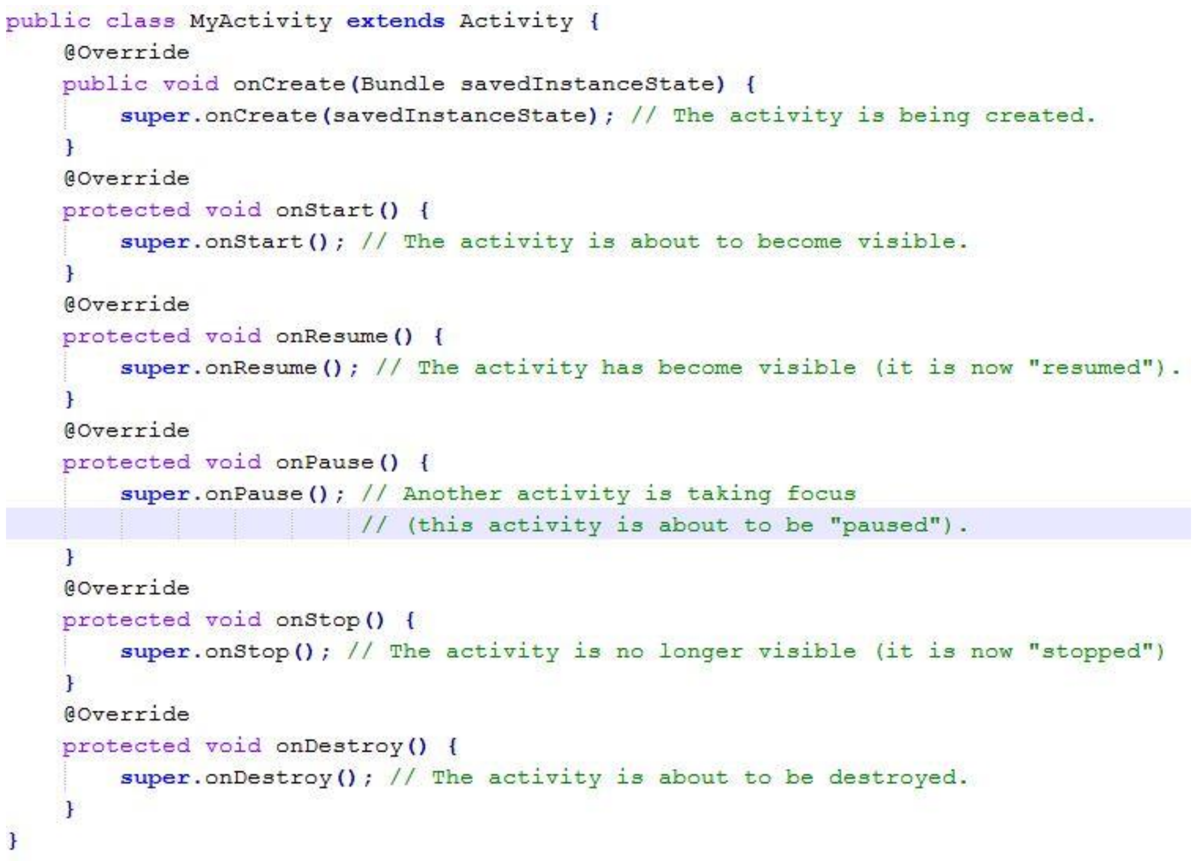
A sample activity is shown in the figure below.



```
public class MainActivity extends Activity {
    Button start;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        start = (Button) findViewById(R.id.start);
        start.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View arg0) {
                startActivity(new Intent(MainActivity.this,
                        AutoprofileActivity.class));
            }
        });
    }
}
```

Figure 7 Sample Activity

4.2.2 Implementing the lifecycle callbacks



```
public class MyActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState); // The activity is being created.
    }
    @Override
    protected void onStart() {
        super.onStart(); // The activity is about to become visible.
    }
    @Override
    protected void onResume() {
        super.onResume(); // The activity has become visible (it is now "resumed").
    }
    @Override
    protected void onPause() {
        super.onPause(); // Another activity is taking focus
                         // (this activity is about to be "paused").
    }
    @Override
    protected void onStop() {
        super.onStop(); // The activity is no longer visible (it is now "stopped")
    }
    @Override
    protected void onDestroy() {
        super.onDestroy(); // The activity is about to be destroyed.
    }
}
```

Figure 8 Lifecycle callbacks

4.3 Intents

An Intent is a messaging object you can use to request an action from another app component. Although intents facilitate communication between components in several ways, there are three fundamental use-cases:

- To start an activity
- To start a service
- To deliver a broadcast

For example, if when working within your own application, you'll often need to simply launch a known activity. You can do so by creating an intent that explicitly defines the activity you want to start, using the class name. For example, here's how one activity starts another activity named SignInActivity.

```
Intent intent = new Intent(this, SignInActivity.class);
startActivity(intent);
```

4.4 Broadcast Receivers

A Broadcast Receiver is a dormant component of the Android system. Only an Intent (for which it is registered) can bring it into action. The Broadcast Receiver's job is to pass a notification to the user, in case a specific event occurs. Using a Broadcast Receiver, applications can register for a particular event. The simplest way to monitor device state changes is to create a BroadcastReceiver for each state you're monitoring and register each of them in your application manifest. Then within each of these receivers you simply reschedule your recurring alarms based on the current device state.

A broadcast receiver is implemented as a subclass of BroadcastReceiver and each broadcast is delivered as an Intent object.

4.5 Services

Service is a fundamental component in android. Many a time, applications will need to run processes for a long time without any intervention from the user or very rare interventions. These background processes need to keep running even when the phone is being used for other activities. Typically a user interacts with a service through an activity that has a UI. The service by itself “notifies” the user in case of any need for user intervention.

To create a service, you must create a subclass of Service (or one of its existing subclasses). In your implementation, you need to override some callback methods that handle key aspects of the service lifecycle and provide a mechanism for components to bind to the service if appropriate.

Chapter 5 - Implementation

Once the user installs the application any number of profiles can be scheduled according to their wish. Also, users can define how often and how long they would like to activate the selected profile. Auto Profile Switcher will recognize their profile based on the current time and it will apply the right profile at the right time according to the given settings. For instance, they can save a home profile and a work profile, and the application will automatically switch from one to other automatically, without any user action.

Also, the application allows users not to bother about the calls from important persons when they are in some important meeting or busy. Then automatically profile is activated as scheduled and sends a respective alert to the caller according to the profile activated. Auto Profile application is user-friendly and easy to use.

The Auto Profile application is developed using Eclipse IDE. ADT plugin for eclipse is installed for Android development environment. The user interface is developed using XML and the business logic is written in Java. Business logic connects to the SQLite database for transactional data. LogCat is used to print the stack traces whenever necessary. The total lines of code written for this application are 3000 which includes Java and XML files.

5.1 Modules

The different modules in the Auto Profile application development are as follows.

- Welcome Screen
- Auto Profile
- Database
- Deletion
- Profile Service
- Message Service

Welcome Screen

Whenever the application starts for the first time a screen which gives the information of the application is popped up. This is the starting activity of the application. This is declared in the manifest file as the Launcher activity. A button is provided which when clicked open the other activity which deals with the profiles. The corresponding XML file in the layout folder takes care of the GUI of this screen.

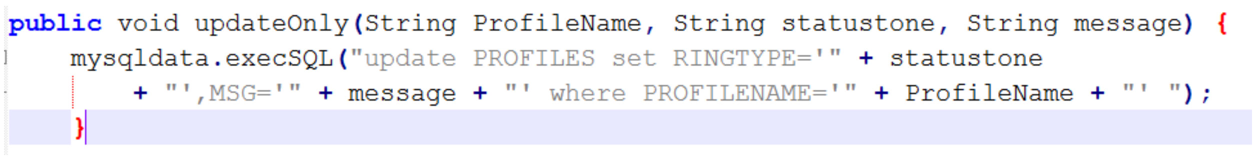
Auto Profile

This is the important module that deals with adding a profile, presenting in a list view and updating it. Also, it deals with restarting the application whenever the mobile is switched off and then on. There are three activities in this module which deals with the above operations that are a screen for presenting all the profiles along with their profile times, a screen with fields which helps user to insert data to create a profile and the final activity which helps users to update the respective profile when clicked on it from a given list. Also, there are respective XML files which are responsible for the user interface.

The first activity is the one which presents all the profiles created using a list view. It has two buttons which help the user to add a profile as well as active the application after the mobile is switched on from a switched off state. When the add profile button is clicked with the help of intents the add profile activity is started and the respective screen shows up. It has two fields to add the profile name as well as the message for that profile. These are carried on to the profile update activity with the help of shared preferences. Once the profiles are added they are visible on the main screen with the profile times not updated. By long pressing the profile name or just by clicking it the update profile activity is started which uses the profile name and message for that profile and displays along with the other fields to enter duration and the ring mode (Silent, Vibrate, Normal) option. The ring mode option is selected using a radio button. The profile is checked against many validations and the profile is then added or updated in the database. Also, this update starts the background Profile service which manages the profile activation.

Database

This module totally deals with database aspect of the Auto Profile application. The database stores all the information related to a profile. Once the profile is updated then it is inserted into the database. The Auto Profile application database consists of only one table which maintains the name of the profile, message associated with the profile, the ring mode for that profile along with the From and To time for that profile. Also the database is hit many times by the background service activated by the user to check if any profile time matches the current time and if there is one, then the mobile ring mode changes to the specified one automatically. A sample method to use the database is shown below.



```
public void updateOnly(String ProfileName, String statustone, String message) {
    mysqldata.execSQL("update PROFILES set RINGTYPE='" + statustone
        + "',MSG='" + message + "' where PROFILENAME='" + ProfileName + "' ");
    }
```

Figure 9 Sample Database method

Whenever the user wants to change only the ring mode and the respective message of a profile the above method is called the respective ring mode and the message of that profile is updated in the database as shown in the above figure.

Deletion

This module deals with the deletion of the profiles as required. The application provides the user two options for deleting the profiles and stopping the background services. The delete options are bound to the menu option of the mobile. Whenever the menu option on the mobile's keyboard is pressed then the two buttons pop up which allows the user to delete all profiles at a time or a specific profile. The deletion of the specific profiles is dealt with an Activity which displays all the profiles in a list view and once clicked on a profile it will be deleted from the database and that specific service is stopped in the backend.

Profile Service

A Service is an application component that can perform long-running operations in the background and does not provide a user interface. This module deals with the scheduling of the profiles from time to time as created by the user. The service is started from the update activity whenever the update button is pressed. Below is the code in the update activity which starts the service with the help of an intent.

```
startService(new Intent(UpdatePro.this, MyProfileService.class));
```

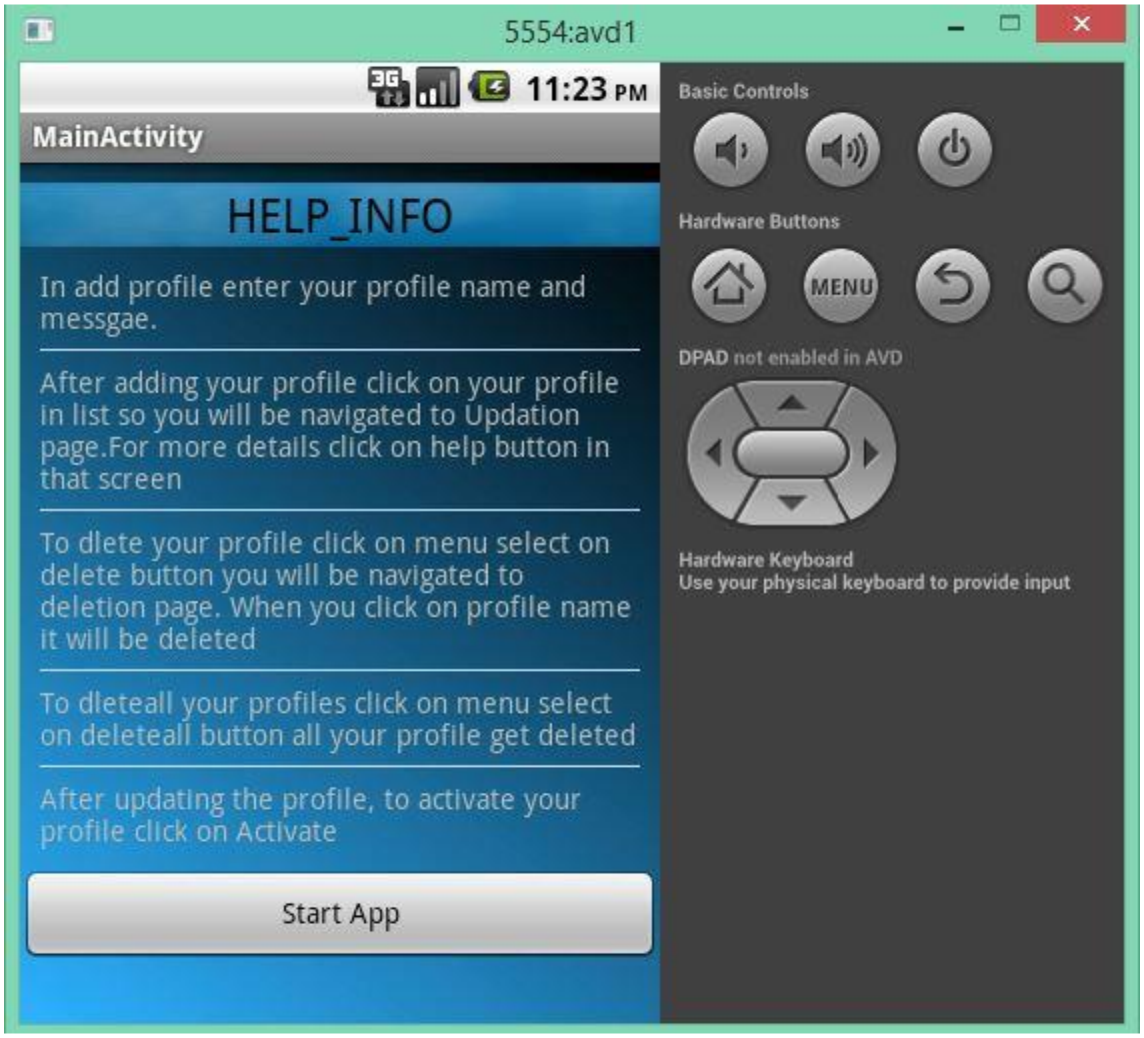
The service hits the data base for every 2 seconds and checks if there is any profile in the database that overlaps with the current system time. If there is any such profile then the respective ring type is retrieved and is assigned to an object of the audio manager. The system ring mode is changed to the respective profile mode accordingly. This happens automatically without any user interaction.

Message Service

This module deals with the messages section of the Auto Profile application. The user sets a specific message for a specific profile and this message will be sent out to the caller. There is a background receiver running all the time which responds to the systemwide broadcast announcements. Whenever user receives a call, the broadcast receiver checks the state of the system using TelephonyManager and if the state is idle, then the calling number is obtained. With the help of an intent this number will be passed to a new service which will be able to send the respective message to the caller. The message service with the help of current system time when the call was received, checks the database if there is any profile active at that point of time. If yes, the respective message will be retrieved and sent to the caller.

5.2 Output Screens

Welcome Screen



HELP_INFO

In add profile enter your profile name and messgae.

After adding your profile click on your profile in list so you will be navigated to Updation page.For more details click on help button in that screen

To dlete your profile click on menu select on delete button you will be navigated to deletion page. When you click on profile name it will be deleted

To dleteall your profiles click on menu select on deleteall button all your profile get deleted

After updating the profile, to activate your profile click on Activate

Start App

Figure 10 Welcome Screen

The above screen pops up for the first time whenever the application is opened. It gives the user information about using the application and details of all the options available to him. Once the Start App button is clicked the below screen opens.

Add Profile

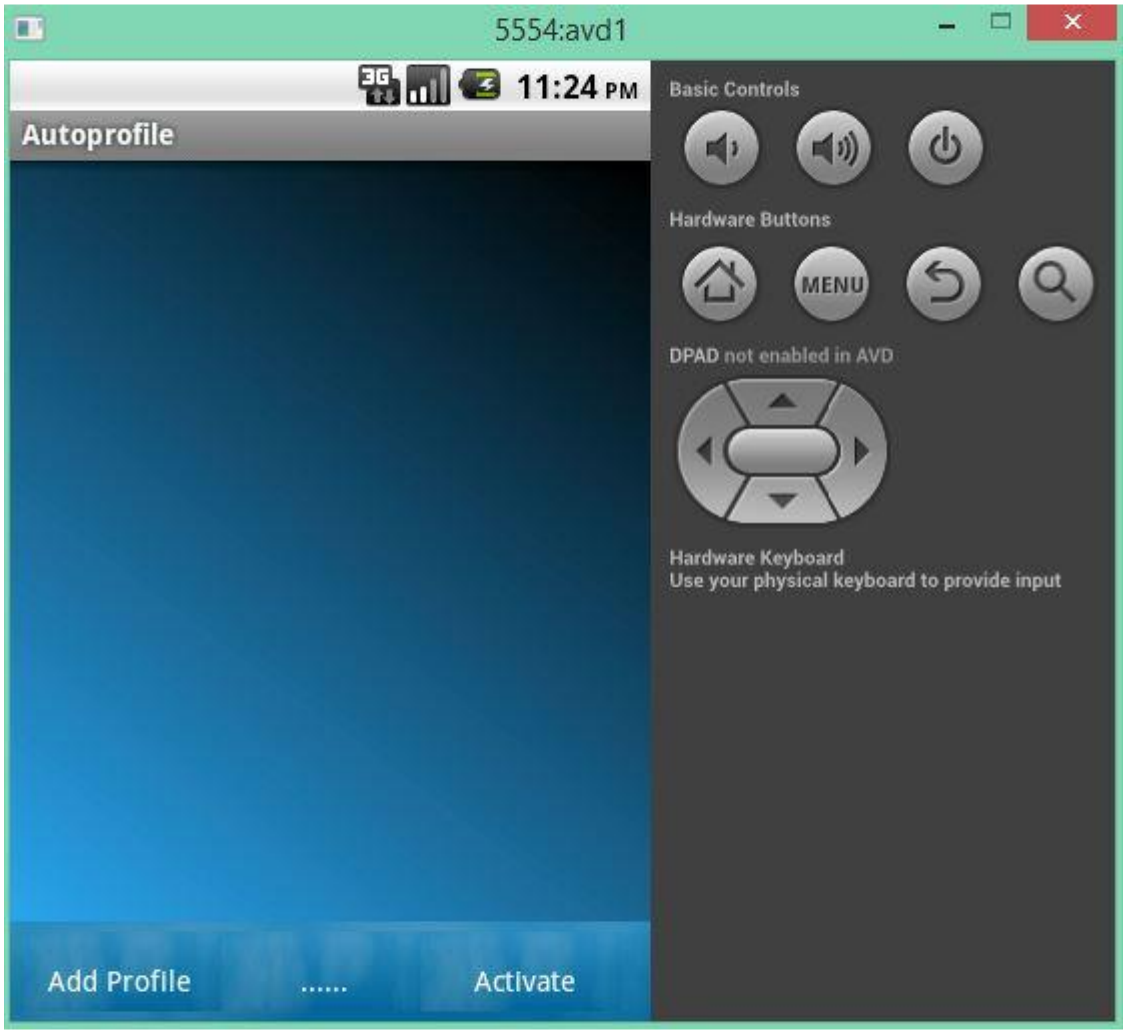


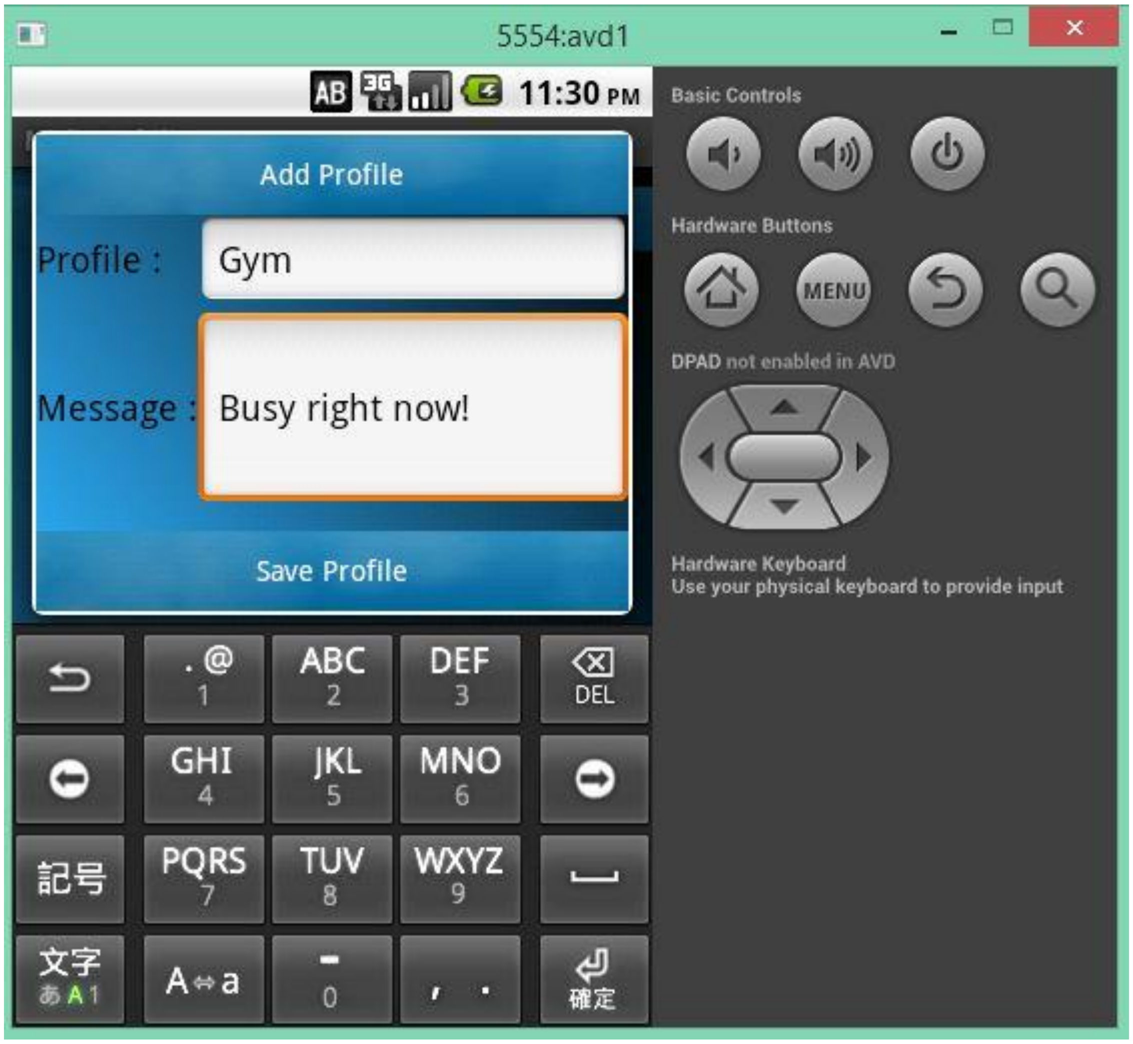
Figure 11 Add Profile

This screen is the one which allows the users to create the custom profiles by clicking the Add Profile Button.

The Activate button is very crucial in situations when the phone is switched off. The background services stops when the mobile phone switches off and hence the service need to be started back. So after the mobile phone is turned on the user must click the Activate Button to restart the service.

Once the Add Profile button is clicked the below screen opens.

Add Profile



Profile : Gym

Message : Busy right now!

Figure 12 Add Profile Screen

This screen allows user to add a custom profile with a unique profile name and the respected message. Once the profile is saved the following screen shows up.

Auto Profiles

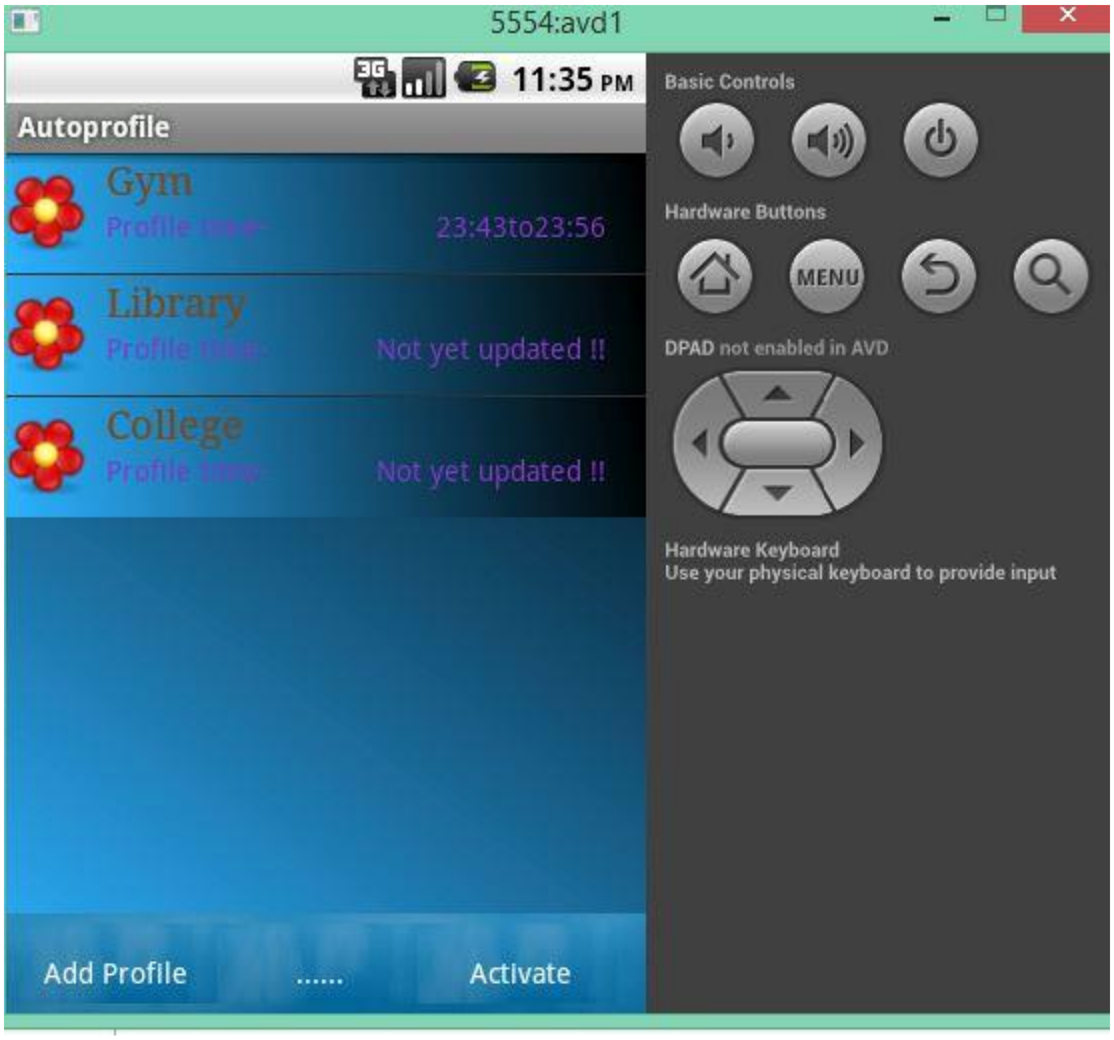
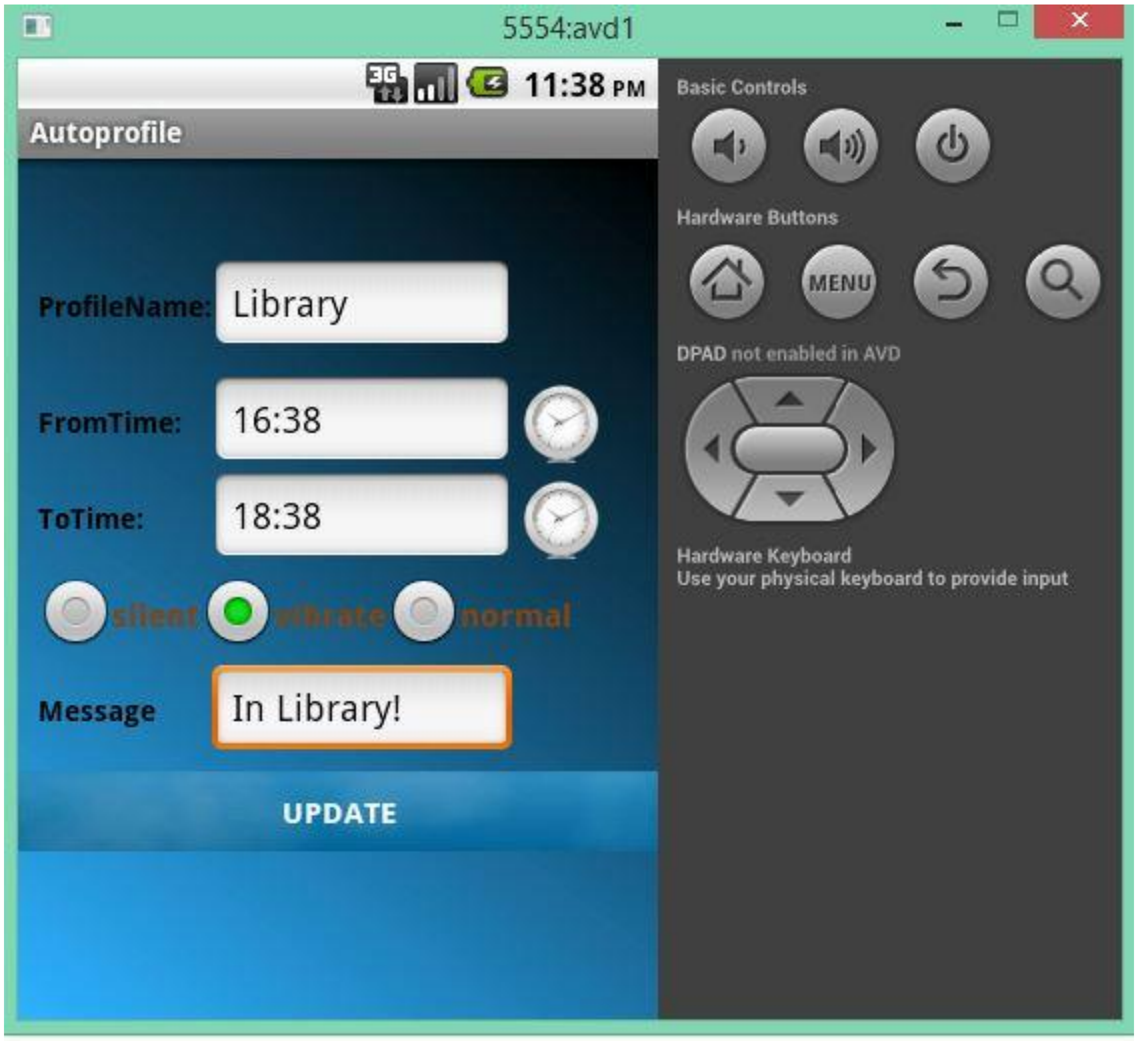


Figure 13 Auto Profiles

This screen shows the list of all the profiles added by the user. The profiles which are scheduled shows from time and to time whereas the profiles that are not updated displays the message that the profile is not yet updated.

By just clicking on a profile or by long press the update screen opens as shown below.

Update Profile



ProfileName: Library

FromTime: 16:38

ToTime: 18:38

silent [✓ vibrate] normal

Message In Library!

Figure 14 Update Profile

This update screen allows user to schedule the profile and edit the message. It allows user to set the required ring mode as well. Many validations are performed in this screen like both from and to times need to be given and also to time cannot exceed from time. One of such validation screen is shown below. When the clock button is pressed the following dialog box opens as below.

Set Times

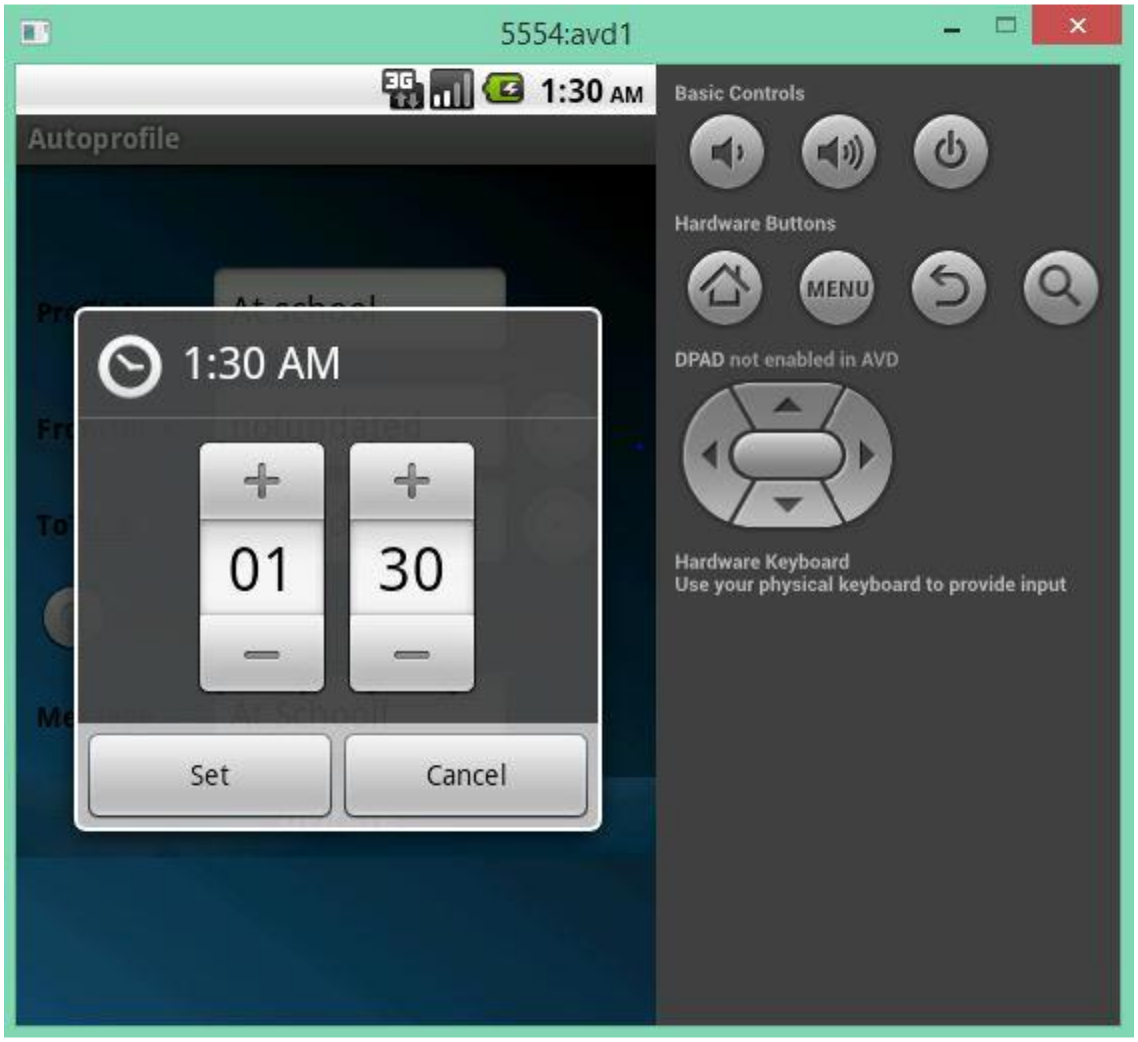
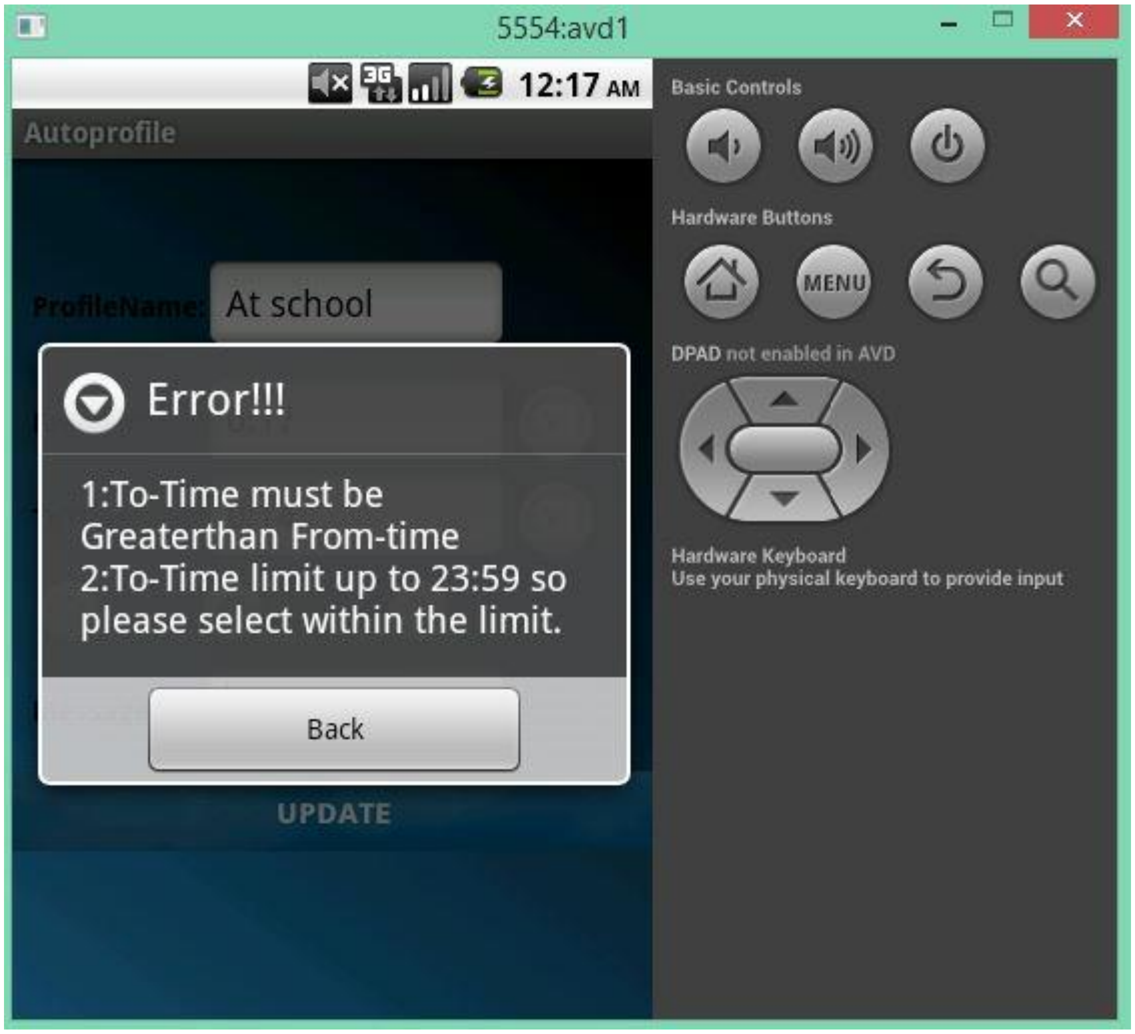


Figure 15 Set Times Dialog box

The above screen opens when the clock image button is clicked by user to enter from time or to time.

Validation Dialog Box



Error!!!

1:To-Time must be Greaterthan From-time
2:To-Time limit up to 23:59 so please select within the limit.

Figure 16 Validation Dialog Box

The above error screen suggests user to make sure that To time must be greater than From time to set up a correct profile.

Delete Single Profile

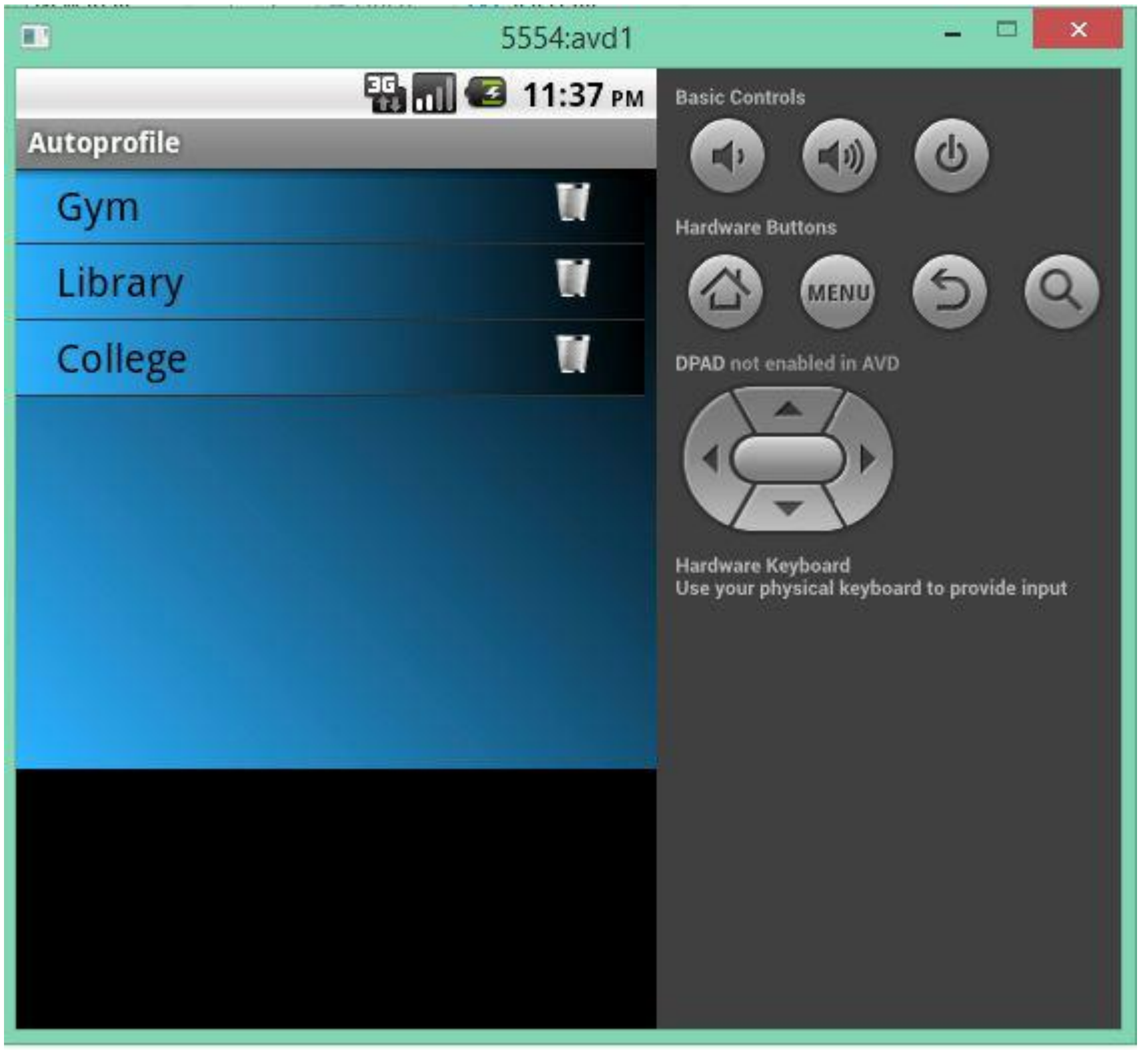


Figure 17 Delete Single Profile

By clicking the menu on the mobile device and then pressing the delete button, the above screen pops up which allows the user to delete a specific profile. The specific profile is deleted when clicked on the bucket image.

Delete All

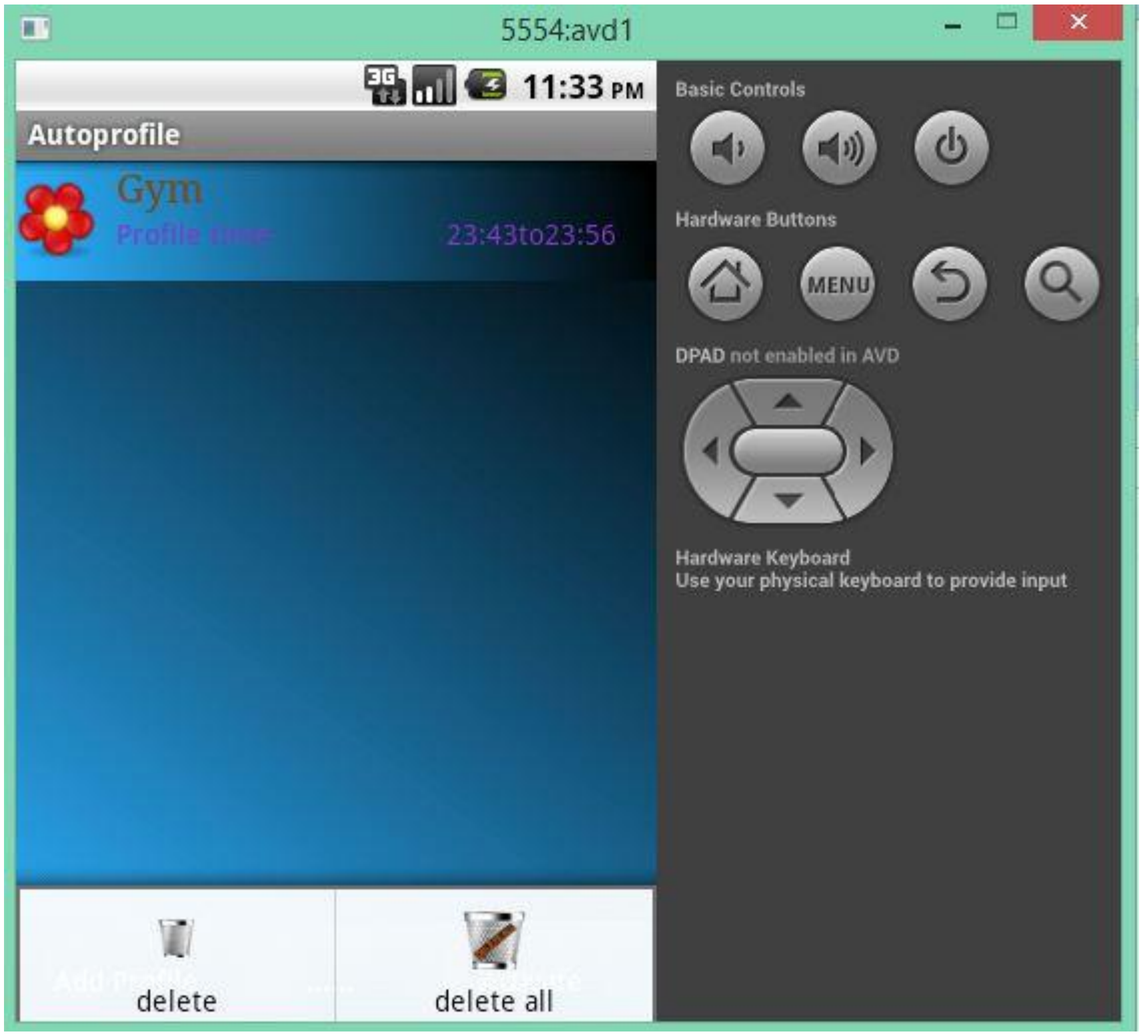


Figure 18 Delete All Profiles screen

Once the menu option is pressed by the user on the mobile device, the above screen opens and allows user to delete all the profiles at a time by pressing the delete all image. This in return stops the running service.

Chapter 6 - Testing

Software testing [8] is a critical element of software quality assurance and represents the ultimate review of specification, design and code generation. The following are the objectives of the testing.

- To ensure that during operation, the system will perform as per specification.
- To make sure, that system meets the user requirements during operation.
- To make sure that during the operation, incorrect input, processing and output will be detected.
- To see that when correct inputs are fed to the system the outputs are correct.
- To verify that the controls incorporated in the same system as intended.
- Testing is a process of executing a program with the intent of finding an error.
- A good test case is one that has a high probability of finding an as yet undiscovered error.

A successful test is one that uncovers an as yet undiscovered error. The software developed has been tested successfully using below testing strategies and any errors that are encountered are corrected and again the part of the program or the procedure or function is put to testing until all the errors are removed.

6.1 Unit Testing

Unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. Unit testing is done for the verification of the code produced during the coding phase and the goal is to test the internal logic of each module. In this project, the unit testing is done during the coding phase of each individual unit to make sure whether the functions are working properly or not.

Sr.no	Test Case	Expected result	Pass/Fail
1	Ask for permissions the application must have	Ask permissions of user to allow application to read phone state, send sms.	Pass
2	Display of App icon after installation	The inserted launcher icon need to be displayed with the application name.	Pass
3	On welcome screen	Display the Help information and the button to enter into the application	Pass
4	On click of Start App button	The Application screen which has list of profiles need to be displayed	Pass
5	On click of Add profile button	The screen which asks for the profile name and message must be opened.	Pass
6	On click of Save Profile	The profile name and message are inserted into data base with few blank fields.	Pass
7	On click of any list item in Auto profile screen	The profile update page with all the Edit texts and labels need to be displayed	Pass
8	On click of Timer button image	The current system time is grasped and displayed using a dialog box.	Pass
9	On click of Update button	All the profile information is inserted into the database.	Pass
10	On click of Activate Button	The service restarts once the mobile device is switched off and this button is clicked	Pass
11	On click of Update button on update profile screen.	The service automatically started in background and the ringer mode changes on the time set.	Pass
12	On click of menu option on keyboard	The delete and delete all options show up on the main screen.	Pass

13	On click of the delete button icon.	List all the available profiles with the button image beside them.	Pass
14	On click of bucket button image on delete screen	Delete all the details of a specific profile from the data base.	Pass
15	On click of delete all button	All the profiles are deleted from the database and the empty main screen is displayed.	Pass
16	On receiving call at time a profile set.	If the call is not lifted or cut, the respective message will be sent to the concerned calling person.	Pass
17	On message sent	Message sent toast message must be displayed to the user.	Pass
18	Background services running as expected.	Profiles changes from one to other as expected	Pass

Table 1 Unit Testing

6.2 Integration Testing

All the above tested individual units are combined as required into subsystems, which are then tested again. The goal of this testing was to see if the modules are properly integrated, and the emphasis being on the testing interfaces between the modules. Focus was made on clicking of the back button on the device and verifying that the navigation was intact. Also, the goal was to verify that the database transactions are doing the right job for any related user actions.

6.3 Compatibility Testing

The Auto profile application is run on various AVDs and made sure that the interface is adapt to various screens. Also, it is installed on many android devices such as Nexus 4, LG G2 and HTC One mini, Samsung S3 and verified that the application is user-friendly in both portrait and landscape modes.

Chapter 7 - Conclusion

First of all I gained many skills in the Android development as this is my first attempt towards it. I also learned a new language that is XML, a language I didn't know at all earlier. Also, this project allowed me to use my SQL skills acquired during my studies.

Auto profile is an excellent application that helps us in our daily life. We might forget to mute our phone before entering into any important meeting and lose the impression of the delegates sometimes. So using this application the user can focus on his daily activities without worrying to monitor his mobile device. Once he sets all his profiles according to the schedule the application takes care of it and changes the profiles according to the time. Also, the application notifies the caller that the user is at an important task and makes him wait before calling again.

The application was developed using most of the steps in Software Development life cycle like Requirements phase, Implementation phase, and the most important testing phase.

Chapter 8 - Future Work

The application Auto profile can be enhanced in a number of ways like

- The application can be developed or enhanced in such a way that the user is allowed to create custom profiles which allows calls from very important persons and which blocks the calls of other people.
- The application can be developed differentiating the phone calls and the normal messages.
- The application can be enhanced to notify the user regularly that he missed a call from an important person at a certain time.

Bibliography

1. Android OS. [Online] [Cited: February 20, 2015.] http://www.tutorialspoint.com/android/android_architecture.htm
2. Android Architecture. [Online] [Cited: February 22, 2015.] http://android-app-tutorial.blogspot.com/2012/08/architecture-system-application-stack.html#.VSwS4fnF-So
3. Class Diagram. [Online] [Cited: February 25, 2015.] http://en.wikipedia.org/wiki/Class_diagram
4. AndroidManifest.xml [Online] [Cited: March 05, 2015.] http://developer.android.com/guide/topics/manifest/manifest-intro.html
5. Activities. [Online] [Cited: March 10, 2015.] http://developer.android.com/training/basics/activity-lifecycle/starting.html
6. Broadcast Receivers. [Online] [Cited: March 14, 2015.] http://stackoverflow.com/questions/5296987/what-is-broadcastreceiver-and-when-we-use-it
7. Services. [Online] [Cited: March 20, 2015.] http://developer.android.com/guide/components/services.html
8. Testing. [Online] [Cited: March 28, 2015.] http://en.wikipedia.org/wiki/Software_testing