A Project Report

on

SMART SOUND SWITCH

Submitted in partial fulfillment of the requirements for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

by

Bapanapalli Manisha (16WH1A1207)

Gandra Samskruthi Rao (16WH1A1219)

Guttikonda Madhavi (16WH1A1225)

Uradi Bhavani (16WH1A1259)

Under the esteemed guidance of

Dr. Aruna Rao S.L
Professor & HoD

Department of Information Technology

BVRIT HYDERABAD COLLEGE OF ENGINEERING FOR WOMEN

(Accredited by NBA & NAAC '**A**' Grade, Approved by AICTE, and
Affiliated to JNTUH, Hyderabad)

Bachupally, Hyderabad – 500090

April 2020

DECLARATION

We hereby declare that the work presented in this project entitled **"SMART SOUND SWITCH"** submitted towards completion of the major project in IV year of B.Tech IT at BVRIT HYDERABAD COLLEGE OF ENGINEERING FOR WOMEN, Hyderabad is an authentic record of our original work carried out under the esteem guidance of **Dr. Aruna Rao S.L, Professor & HoD**, IT department.

Bapanpalli Manisha
(16WH1A1207)

Gandra Samskruthi Rao
(16WH1A1219)

Guttikonda Madhavi
(16WH1A1225)

Uradi Bhavani
(16WH1A1259)

BVRIT HYDERABAD COLLEGE OF ENGINEERING FOR WOMEN

(Accredited by NBA & NAAC 'A' Grade, Approved by AICTE, and Affiliated to JNTUH, Hyderabad)

Bachupally, Hyderabad – 500090

Department of Information Technology

Certificate

This is to certify that the Project report on "**Smart Sound Switch**" is a bonafide work carried out by **Bapanpalli Manisha (16WH1A1207), Gandra Samskruthi Rao (16WH1A1219), Guttikonda Madhavi (16WH1A1225), Uradi Bhavani (16WH1A1259)** in the partial fulfillment for the award of B.Tech degree in **Information Technology, BVRIT HYDERABAD COLLEGE OF ENGINEERING FOR WOMEN, Bachupally, Hyderabad**, affiliated to Jawaharlal Nehru Technological University, Hyderabad under my guidance and supervision.

The results embodied in the project work have not been submitted to any other University or Institute for the award of any degree or diploma.

External Examiner

Internal Guide & HoD
Dr. Aruna Rao S L
Professor and HOD
Department of IT

Acknowledgements

We would like to express our sincere thanks to **Dr. K. V. N. Sunitha, Principal**, BVRIT HYDERABAD, for providing the working facilities in the college.

Our sincere thanks and gratitude to our internal guide **Dr**. **Aruna Rao S L, Professor and HoD**, **Department of Information Technology,** BVRIT HYDERABAD for all the timely support and valuable suggestions during the period of our project and for her constant guidance, encouragement and moral support throughout the project.

Finally, we would also like to thank our Project Coordinator, **L. Naveen Kumar**, **Assistant Professor, Department of Information Technology**, all the faculty and staff of Information Technology Department who helped us directly or indirectly, parents and friends for their cooperation in completing the project work.

Bapanpalli Manisha
(16WH1A1207)

Gandra Samskruthi Rao
(16WH1A1219)

Guttikonda Madhavi
(16WH1A1225)

Uradi Bhavani
(16WH1A1259)

ABSTRACT

Smart Sound Switch is a mobile application which mainly focuses on the automation of sound mode switching based on locations when a call is received. There are places like temples, churches, mosques, public meetings (assembly, parliament), hospitals, funeral grounds, theatres, your workplace, schools, colleges, where it is a basic mobile etiquette of smartphone user to keep the mobile phones in silent mode rather than put in awkward situations because of our ignorance and overlook. To support all the users on sound mode management, we have come up with an innovative solution to control sound mode switching. On the reception of a call, the location of the user is checked against the chosen locations and the mode is switched automatically as per the preferences set. The application includes unique features such as selective inclusions/exclusions of their point of interest, forced sound mode resetting when multiple calls are received and for events that are stored in Google/Outlook calendar. The technology stack used is Android Studio and Google places (API).

LIST OF FIGURES

LIST OF ABBREVATIONS

Term/Abbreviation	Definition
GPS	Global Positioning System
API	Application Programming Interfaces
LBS	Location Based Service
ADT	Android Development Tools
LSU	Location Server Update
SDK	Software Development Toolkit
JVM	Java Virtual Machine

TABLE OF CONTENTS

1. INTRODUCTION

1.1 OBJECTIVE

To build an automated Android application for automatic profile switching as per location and time. There are many places like Hospitals, Petrol pumps, Universities, Corporate offices etc. where it is clearly mentioned, “KEEP YOUR MOBILE PHONES SILENT!!” Many times people forget to switch the mobile to the “Silent Mode” which is not feasible every time like in an important meeting, lectures etc. This application provide near about completely automated profile switching according to location. This application will enable the device to switch to the ‘Silent Mode’ in locations like Hospitals, Major Corporate offices, Universities, Well known Educational Institutions, Petrol pumps, Government offices etc. based on your preferences set earlier manually. It also takes the data such as meetings time, date, location etc. from the email and set the preferred mode automatically. This application is user friendly as it allows the user to set the user defined profile mode with the help of user defined settings.

1.2 PROBLEM IN EXISTING SYSTEM

There are various android applications to change the profiles based on time. The application switches the profile to not only ringer mode but the entire sound profile is auto adjusted smartly according to the time. There are limitations in the android applications such as the user has to set the location manually & then it switches to the profiles based on specified time which does not make it completely automated.

1.3 SOLUTION

SMART SOUND SWITCH will enable the device to switch to the Silent Mode in locations like Hospitals, schools, colleges, Universities, offices etc. as per customized by the user. The user just needs to set the required locations which will be stored in SQLite and compared with GPS coordinates and automatically switch the profiles whenever a call is received. If being on the silent mode say, in a meeting, a call comes from a particular number, the profile will change automatically to general mode after receiving more than 3 calls, and back to the silent mode after the call is been attended.

The user can contact them back later without being disturbed. In User- Defined Switching Mode user set location that gets stored in the SQLite database which is already present in Android Devices. The application will use GPS Service provided by GPS Satellites for finding locations. In profile switching operation application actually switch the ringer mode of profile.

The application will use GPS Service provided by GPS Satellites for finding locations. In profile switching operation application actually switch the ringer mode of profile. Here user can choose among Silent, Vibrate and ringer mode for switching purpose. There is a provision made to neglect the calls while on the silent profile to avoid the disturbance. Only Calls from the emergency numbers stored by the users will be allowed to be attended. Thus, PROFILE MANAGEMENT SYSTEM plans to achieve the following objectives:

- Easy to use
- Automated profile switching
- Accuracy
- Increase usability
- User-friendly

1.4 FEATURES

- Wide range of default locations, such as educational institutions, medical complexes, government and corporate offices etc. based on the user preferences.
- Provision for adding those locations in silent zone which are not covered in Default Silent Zone.
- User-defined accuracy setting for user defined locations.
- User-defined locations are stored in device's SQLite database and not in GPS Server Database hence GPS Server Database is not get disturb for adding new user-defined location or updating existing user-defined locations.

- User can set the profile mode volume based on their preferences and can vary in different locations.
- He can also allow the access of calendar to the application. So, the application can automatically get the details of the meeting and events that are scheduled in the future and change the profile mode accordingly.
- Volume can not only be set to silent, ringer and vibration but also can be changed to levels of 1 to 7.
- We can add the current location or can select any particular location. We have an option to edit the specifications even after creating an event.

2. LITERATURE SURVEY

2.1. INFORMATION GATHERING:

2.1.1. ANDROID:

Android delivers a complete set of software for mobile devices: an operating system, middleware and key mobile applications. Android was built from the ground-up to enable developers to create compelling mobile applications that take full advantage of all a handset has to offer. It was built to be truly open. Android is built on the open Linux Kernel.

Android support LBS Application Programming Interfaces (APIs). Location service allows finding out the device current location. The application can request for periodic update of the device location information. The application can also register a intent receiver for proximity alerts like when the device is entering and existing from an area of given longitude, latitude and radius.

On a basic level, android is a distribution of Linux that includes a Java Virtual Machine (JVM), with Java being the preferred programming language for most Android applications. The Android Software Development Kit (SDK) includes a debugger, libraries, a handset emulator, documentation, sample code and tutorials. Android‘s official integrated development environment is Eclipse using the Android Development Tools (ADT) plug-in. SQLite database support is integrated into the Android platform. The ADT plug-in includes an Android emulator that allows for the simulation of GPS and Wi-Fi.

2.1.2. GPS:

The Global Positioning System (GPS) is a space based satellite navigation system that provides location and time information in all weather conditions, anywhere on or near the Earth where there is an unobstructed line of sight to four or more GPS satellites.

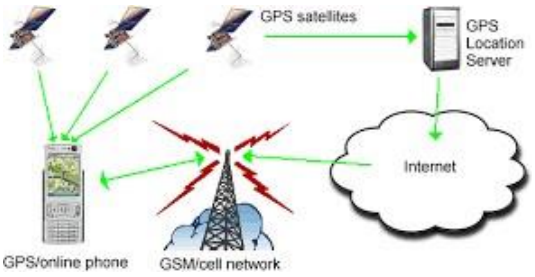


Figure 2.2.2 Global Positioning System

GPS Architecture The system provides critical capabilities to military, civil and commercial users around the world. It is maintained by the United States government and is freely accessible to anyone with a GPS receiver. GPS is often used by civilians as a navigation system [4]. On the ground, any GPS receiver contains a computer that "triangulates" its own position by getting bearings from at least three satellites. The result is provided in the form of a geographic position - longitude and latitude - to, for most receivers, within an accuracy of 10 to 100 meters. Software applications can then use those coordinates to provide driving or walking instructions.

Getting a lock on by the GPS receivers on the ground usually takes some time especially where the receiver is in a moving vehicle or in dense urban areas. The initial time needed for a GPS lock is usually dependent on how the GPS receiver starts. The receiver has a general idea of which satellites to look for because it knows its last position and the almanac data helps identify which satellites are visible in the sky. This takes longer than a hot start but not as long as a cold start. The GPS receiver has to attempt to lock onto a satellite signal from any available satellites, basically like polling, which takes a lot longer than knowing which satellites to look for. This GPS lock takes the longest. In an attempt to improve lock times, cell phone manufacturers and operators have introduced the Assisted GPS technology, which downloads the current ephemeris for a few days ahead via the wireless networks and helps triangulate the general user‘s position with the cell towers thus allowing the GPS receiver to get a faster lock at the expense of several (kilo) bytes[5].

2.1.3. CALENDAR SERVICES:

This service allows a script to read and update the user's Google Calendar. This class provides direct access to the user's default calendar, as well as the ability to retrieve additional calendars that the user owns or is subscribed to.

3. REQUIREMENT SPECIFICATION

3.1.1 SOFTWARE REQUIREMENT

- Operating system: Windows 7/XP/8.0/8.1/10
- Coding Language: java jdk/ jre. Android jdk Eclipse (IDE)
- Backend: Java
- Server: MYSQL database
- Google Maps
- Google Calendar or Outlook
- SQLite
- Emulator(Testing on System)

3.1.2 HARDWARE REQUIREMENTS

- System: 2GB ram 64bit processor
- Hard Disk: 80 GB
- Mouse: Optical Mouse
- Keyboard: 101 Keys
- Android Device: 4.0 (Ice-cream sandwich) and above versions

4. SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

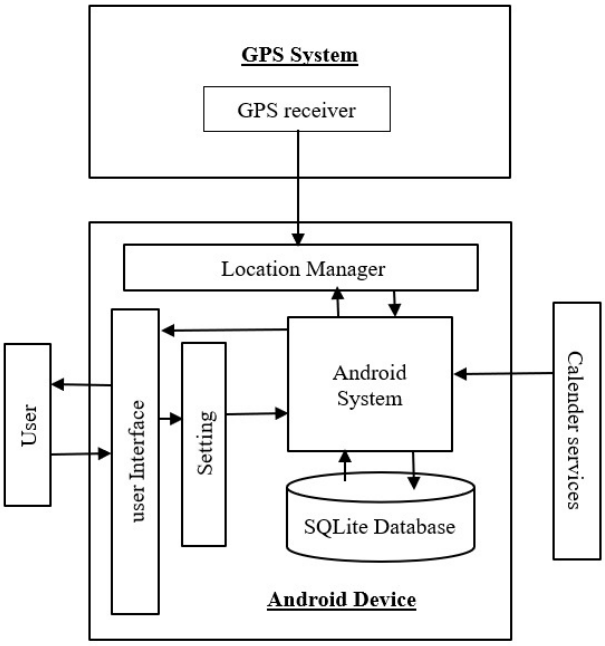


Figure 4.1 : System Architecture

In the Android Device, the Location Manager is an interface between Android Device and GPS System. Using Forward Geocoding method Android System will get the co-ordinates of Android Device from GPS Satellite through Location manager. Then those co-ordinates will send to GPS Server Database to get name and address of location this method is known as Reverse Geocoding. After getting name and address of location, the Android System will check that whether the received address is belongs to Default Silent Zone or not. If device is in Default Silent Zone then Android System will switch sound profile ringer mode to Silent or Vibrate only as per settings.

Using User Interface user can store location information i.e. co-ordinates in the SQLite Database. While storing the location user can give any name for the particular location, also he will able to choose mode i.e. Silent or Vibrate Only and Activation status. User will also able to change settings for User-Defined as well as Default Switching and turn on/off the application through Settings.

4.2. WORKING

4.2.1 USER-DEFINED MODE SWITCHING:

If current location is not belong to Default Silent Zone then and then only device work in User-Defined Mode. In User-Defined Mode flow of processes will be as follows.

1) Finding location in SQLite Database: Coordinates of current location find in Default Mode will be used in User-Defined Mode also. Using those coordinates device checks the entry made by user for that coordinates in its SQLite Database. If device founds entry in SQLite Database then device is in Silent Zone else not.

2) Checking status of location: If device is in Silent Zone then device checks the status for that user-defined location, whether it is activated or not. If it is activated then device is in Activated Silent Zone else not.

3) Switching sound profile accordingly: If location is not belongs to the Activated Silent Zone then maintain ringer mode as 'General' else switch ringer mode as per user setting to Complete 'Silent Mode' or 'Vibrate Only Mode'.

Accuracy for switching can also be controlled by user at the time setting new location to User-Define Silent Zone or updating existing location in User-Defined Silent Zone.

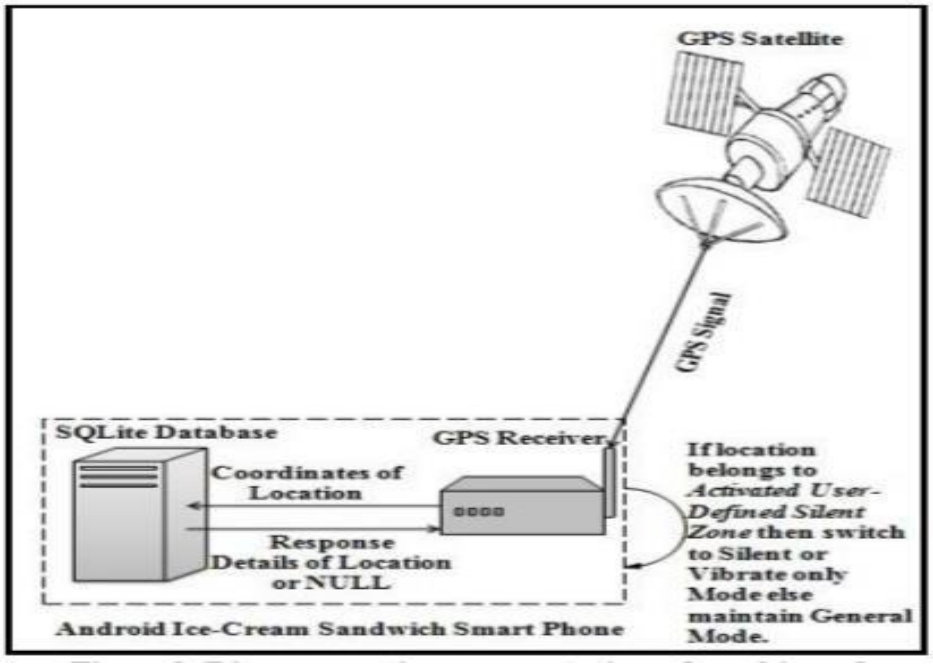


Fig.4.2 Diagrammatic representation of working of application in User-Defined Mode

The GPS Satellite continuously transmits the signal containing information about receiver's location. Using this information GPS receiver calculates coordinates of location. The GPS Server Database stores the information of locations such as coordinates of location (i.e. Longitude, Latitude, and Altitude) and name and address of that location. If device is in Silent Zone then Android System will switch sound profile ringer mode to Silent or Vibrate only as per settings. Android System will check for user defined Silent Zone in SQLite Database which is already present in Android Device. If location does not belongs to Silent Zone then switching will not takes place. User can add location for automatic profile switching.

4.2.2 USER SETTINGS

User Settings helps user to control the switching process in both Default Mode and User-Defined Mode. It provides following functionalities.

1) Turn on / off Application: This function provides the facility to turning on / off this application to the user. This application is capable to run in background process. After installation, if user don't want to use the application then he can turn off the application and whenever he want he can turn it on again.

2) Enable / Disable Default Mode: Using this function user can decide that in his running application whether he wants Default Mode to be activated or not.

3) Enable / Disable User-Defined Mode: Using this function user can decide that in his running application whether he wants user defined Mode to be activated or not.

4) Silent / Vibrate/ Flight Default Mode: User can set switching mode (i.e. Complete Silent or Vibrate only or Flight) for Default Silent Zone using this function.

5) Silent / Vibrate all User-Defined Mode: User can set switching mode (i.e. Complete Silent or Vibrate only) for User-Defined Silent Zone using this function.

6) Set new User-Defined Mode: This function helps user to add new locations (Not more than 20) in User-Defined Silent Zone (i.e. in SQLite Database).

7) Update User-Defined Mode: This function helps user to update (i.e. edit or delete) existing locations in User-Defined Silent Zone (i.e. in SQLite Database).

5. MODULES

SYSTEM MODULE

Our project is about providing automated profile switching that will enable the device to switch the 'Sound Mode' based on the data stored by the users in the database as per their requirements.

- Location module
- Profile Changer module
- Logs

5.1.1 Module - 1 : LOCATION MODULE

This module consists of the Google maps using which the user can set required locations to be used for profile changing. The name for the location and the range of radius is defined along with the coordinates of the locations selected.

- This module consists of the Google maps.
- The user can set required locations to be used for sound mode switching.
- The user can set the category of location(Ex: Hospital, Temple etc.,).

5.1.2 Module - 2 : PROFILE CHANGER MODULE

This module contains locations that are to be activated for the call rejection and profile changing functions. It plays an important role in mapping the groups with the locations selected.

- Using this module the locations are activated to the predefined mode when a call is received.
- Sound mode changing function is also allowed for repeated calls within defined time.
- It plays an important role in mapping the groups with the locations selected.

5.1.3 Module – 3 : LOGS

This module simply helps to keep the track of the user call logs. The type of calls i.e. incoming, outgoing or missed call along with its date and time of calling in saved for the future reference of the user. Logout: This module is being used when the user desires to exit the application. The application stops working and the user needs to log in again in order to continue using the services.

- This module simply helps to keep the track of the user call logs.
- The type of calls i.e. incoming, outgoing or missed call along with its date and time of calling is saved for the future reference of the user.

6. IMPLEMENTATION

6.1. Code to store and activate the location, calendar event and sound mode

```
import android.Manifest;
import android.content.*;
import android.content.pm.PackageManager;
import android.database.Cursor;
import android.graphics.Point;
import android.location.Location;
import android.media.AudioManager;
import android.net.Uri;
import android.os.Build;
import android.os.Environment;
import android.provider.CalendarContract;
import android.support.annotation.RequiresApi;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.*;
import android.widget.*;
import android.provider.CalendarContract.Calendars;

import java.io.File;
import java.io.IOException;
import java.util.*;

public class MainActivity extends AppCompatActivity implements
View.OnClickListener
{
   private AudioManager amanager;

   private static final int READ_CALENDAR_EVENTS = 113;

   @RequiresApi(api = Build.VERSION_CODES.M)
   @Override
   protected void onCreate(Bundle savedInstanceState)
   {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      i.helper.changeStatusBarColor(MainActivity.this);

      getScreenResolution();
```

```
        Log.v("locationsListUser.size",
String.valueOf(i.helper.locationsListUser.size()));

        addTempLocations();

        setHeader();
        test();
    }

    @Override
    protected void onStop()
    {
        super.onStop();
    }

    private void getScreenResolution()
    {
        Display display = getWindowManager().getDefaultDisplay();
        Point size = new Point();
        display.getSize(size);
        int width = size.x;
        int height = size.y;
        Log.e("Width", "" + width); //1080
        Log.e("height", "" + height); //1776
    }

    private void test()
    {
        Location currentLocation = new Location("Trigeo Technologies");
        currentLocation.setLatitude(17.7133);
        currentLocation.setLongitude(83.3151);

        Location pLocation1 = new Location("KGH");
        pLocation1.setLatitude(17.708967);
        pLocation1.setLongitude(83.306043);

        Location pLocation2 = new Location("MY Home");
        pLocation2.setLatitude(17.749811);
        pLocation2.setLongitude(83.262718);

        float pDistanceToAddress =
i.helper.getDistanceInKMsBetweenTwoLocations(currentLocation, pLocation2);

        android.support.v7.app.AlertDialog.Builder pAlert2 = new
android.support.v7.app.AlertDialog.Builder(this);
        pAlert2.setTitle("Distance of Destination Location");
        pAlert2.setMessage("The shortest distance is " + pDistanceToAddress + " from
Current location");

        pAlert2.setPositiveButton("OK", null);
```

```
        pAlert2.show();
    }

    public void getCalendarEvents()
    {
        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.READ_CALENDAR) !=
PackageManager.PERMISSION_GRANTED)
        {
            ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.READ_CALENDAR},
this.READ_CALENDAR_EVENTS);
        }

        ContentResolver pContentResolver = getContentResolver();

        Cursor pCursor =
pContentResolver.query(Uri.parse("content://com.android.calendar/events"),
                new String[]{ "calendar_id",
                        "title",
                        "description",
                        "dtstart",
                        "dtend",
                        "eventLocation" }, null, null, null);//Keys must give as it is. If we
modify, we can't get events and app will get crash!!!

        int totalEvents = pCursor.getCount();

        pCursor.moveToFirst();

        for (int index = 0; index < totalEvents; index++)
        {
            ACEvent pACEvent = new ACEvent();
            pACEvent.idEvent = pCursor.getInt(0);
            pACEvent.titleEvent = pCursor.getString(1);
            pACEvent.descriptionEvent = pCursor.getString(2);
            pACEvent.dateStart = new Date(pCursor.getLong(3)).toString();
            pACEvent.dateEnd = new Date(pCursor.getLong(4)).toString();
            pACEvent.locationDescription = pCursor.getString(5);

            i.helper.eventsCalendarList.add(pACEvent);

            pCursor.moveToNext();
        }

        pCursor.close();

        Log.e("eventsCalendarList", i.helper.eventsCalendarList.toString());
    }
```

```
public void createTimerToUpdateRingerVolume()
{
   Timer timer = new Timer();

   final int FPS = 100;
   ACTimerTask pACTimerTask = new ACTimerTask(this);
   timer.scheduleAtFixedRate(pACTimerTask, 0, 1000/FPS);
}

public void setHeader()
{
   //TextView txt_left = (TextView)findViewById(R.id.id_txtLeft);
   TextView txt_Header = (TextView)findViewById(R.id.id_txtHeading);
   TextView txt_right = (TextView)findViewById(R.id.id_txtRight);

   //txt_left.setText("Settings");
   txt_right.setText("Settings");
   txt_right.setVisibility(View.INVISIBLE);
}

@Override
public void onClick(View view)
{
   int id = view.getId();

   if (id == R.id.btn_hotel)
   {
      //To delete
      //createTimerToUpdateRingerVolume();

      Log.v("onClick", "btn_hotel");
      i.helper.currentLocation = new Location("Novotel");
      i.helper.currentLocation.setLatitude(17.7110);
      i.helper.currentLocation.setLongitude(83.3160);
   }
   else if (id == R.id.btn_office)
   {
      Log.v("onClick", "btn_office");
      i.helper.currentLocation = new Location("Trigeo Technologies");
      i.helper.currentLocation.setLatitude(17.7133);
      i.helper.currentLocation.setLongitude(83.3151);
   }
   else if (id == R.id.btn_temple)
   {
      Log.v("onClick", "btn_temple");
      i.helper.currentLocation = new Location("Kalimatha Temple");
      i.helper.currentLocation.setLatitude(17.712603);
      i.helper.currentLocation.setLongitude(83.318791);
   }
```

```
        else if (id == R.id.btn_hosp)
        {
            Log.v("onClick", "btn_hosp");
            i.helper.currentLocation = new Location("KGH");
            i.helper.currentLocation.setLatitude(17.708967);
            i.helper.currentLocation.setLongitude(83.306043);
        }
        else if (id == R.id.id_txtRight)
        {
            Log.v("onClick", "navigation 2");

            Intent pIntent = new Intent(MainActivity.this, ACSettingsAct.class);
            startActivity(pIntent);
        }
        else if (id == R.id.btn_ring)  //2
        {
            int mode = AudioManager.RINGER_MODE_NORMAL;
            testRingerMode(mode);
        }
        else if (id == R.id.btn_vibration)  //1
        {
            int mode = AudioManager.RINGER_MODE_VIBRATE;
            testRingerMode(mode);
        }
        else if (id == R.id.btn_silent)  //0
        {
            int mode = AudioManager.RINGER_MODE_SILENT;
            testRingerMode(mode);
        }
        else if (id == R.id.btn_change)
        {
            EditText et_volume = (EditText)findViewById(R.id.et_volumeRange);
            String volume = et_volume.getText().toString();

            AudioManager audioManager = (AudioManager)
getSystemService(getApplicationContext().AUDIO_SERVICE);

            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M)
            {
                if (volume != null)
                {
                    audioManager.setStreamVolume(AudioManager.STREAM_RING,
Integer.parseInt(volume), AudioManager.FLAG_SHOW_UI +
AudioManager.FLAG_PLAY_SOUND);
                }
            }

            et_volume.getText().clear();
        }
    }
```

```
    private void testRingerMode(int mode)
    {
        Log.e("ringerMode:", String.valueOf(mode));

        AudioManager audioManager = (AudioManager)
getSystemService(getApplicationContext().AUDIO_SERVICE);

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M)
        {
            audioManager.setRingerMode(mode);
        }
    }

    private void testStreamVolume()
    {
        Log.e("testStreamVolume", "testStreamVolume");
        AudioManager audioManager = (AudioManager)
getSystemService(AUDIO_SERVICE);
        int maxVolume =
audioManager.getStreamVolume(AudioManager.STREAM_RING);
        audioManager.setStreamVolume(AudioManager.STREAM_RING, maxVolume,
AudioManager.FLAG_SHOW_UI + AudioManager.FLAG_PLAY_SOUND);
    }

    public void addTempLocations()
    {
        Button btn_hotel = (Button)findViewById(R.id.btn_hotel);
        Button btn_office = (Button)findViewById(R.id.btn_office);
        Button btn_temple = (Button)findViewById(R.id.btn_temple);
        Button btn_hospital = (Button)findViewById(R.id.btn_hosp);

        btn_hotel.setOnClickListener(this);
        btn_office.setOnClickListener(this);
        btn_temple.setOnClickListener(this);
        btn_hospital.setOnClickListener(this);
        Button btn_ring = (Button)findViewById(R.id.btn_ring);
        Button btn_silent = (Button)findViewById(R.id.btn_silent);
        Button btn_vibration = (Button)findViewById(R.id.btn_vibration);

        Button btn_change = (Button)findViewById(R.id.btn_change);

        btn_ring.setOnClickListener(this);
        btn_silent.setOnClickListener(this);
        btn_vibration.setOnClickListener(this);
        btn_change.setOnClickListener(this);
    }
}
```

6.2. Code for accessing and retrieving data from calendar events

```
import android.accessibilityservice.AccessibilityService;
import android.app.ActivityManager;
import android.app.AlertDialog;
import android.content.Intent;
import android.location.Address;
import android.location.Geocoder;
import android.location.Location;
import android.provider.Settings;
import android.util.Log;
import android.view.accessibility.AccessibilityEvent;
import com.google.api.client.util.DateTime;
import java.io.IOException;
import java.text.*;
import java.util.*;

import static android.view.accessibility.AccessibilityEvent.TYPE_NOTIFICA
TION_STATE_CHANGED;
public class ACCalendarService extends AccessibilityService
{
   private long mEventTime;
   public ACCalendarService(Context pContext) {
      super();
      Log.e("ACCalendarService", "ACCalendarService");
   }
   public ACCalendarService() {
      super();
   }
   void init(ACCalendarService event) {
      mEventTime = event.mEventTime;
   }
   public long getEventTime() {
      return mEventTime;
```

```
    }
    protected ACCalendarService(Parcel in) {
        Log.e("ACCalendarService", "Parcel");
        mEventTime = in.readLong();
        Log.e("writeToParcel", String.valueOf(mEventTime));
    }
    public void setEventTime(long eventTime) {
        mEventTime = eventTime;
    }
    public static final Creator<ACCalendarService> CREATOR = new
Creator<ACCalendarService>() {
        @Override
        public ACCalendarService createFromParcel(Parcel in) {
            Log.e("ACCalendarService", "createFromParcel");
            return new ACCalendarService(in);
        }
        @Override
        public ACCalendarService[] newArray(int size) {
            Log.e("ACCalendarService", "newArray");
            return new ACCalendarService[size];
        }
    };
    @Override
    protected void onServiceConnected() {
        super.onServiceConnected();
        Log.e("ACCalendarService", "Service Connected");
        Intent intent = new Intent(Settings.ACTION_ACCESSIBILITY_SETTIN
GS);
        startActivity(intent);
    }
    @Override
    public void onAccessibilityEvent(AccessibilityEvent event) {
        Log.e("ACCalendarService", "onAccessibilityEvent 2");
        if (event.getEventType() == TYPE_NOTIFICATION_STATE_CHAN
GED) {
```

```
if(event.getPackageName().toString().equals("com.samsung.android.calendar"))
            {
               String pText = (String) event.getText().get(0);
               Parcelable pParcelable = event.getParcelableData();
               try {
                  if (event.getParcelableData() != null) {
                     Location pLocation = getLocationFromAddress(pText);
                     float pDistanceToAddress = i.helper.getDistanceInKMsBetwee
  nTwoLocations(i.helper.currentLocation, pLocation);
                     android.support.v7.app.AlertDialog.Builder pAlert2 = new
  android.support.v7.app.AlertDialog.Builder(this);
                     pAlert2.setTitle("Distance of Destination Location");
                     pAlert2.setMessage("The shortest distance is " +
  pDistanceToAddress + " from Current location");
                     pAlert2.setPositiveButton("OK", null);
                     pAlert2.show();
                  }
               } catch (IOException e)
                  e.printStackTrace();
               dateConvertDouble(event);
               Long tag_callback_time = System.currentTimeMillis();
            }
         }
         DateTime start = event.getEventTime() getStart().getDateTime();
         if (start == null) {
            start = event.getStart().getDate();
         }
         eventStrings.add(String.format("%s (%s) location: %s",
  event.getSummary(), start, event.getLocation()));
         Log.e("ACCalendarService", "onAccessibilityEvent 3");
      }
      @Override
      public void onInterrupt() {
      }
```

```
    public void dateConvertDouble(AccessibilityEvent event) {
        Date todate = new Date(event.getEventTime());
        long myLong = todate.getTime();
        String pString = DateFormat.getDateInstance().format(todate);
        Log.e("dateTime", pString);
        Calendar calendar = Calendar.getInstance();
        calendar.setTime(new Date(event.getEventTime()));
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd
hh:mm:ss");
        sdf.setTimeZone(TimeZone.getTimeZone("UTC"));
        System.out.println(sdf.format(calendar.getTime()));
        //Here you set to your timezone
        sdf.setTimeZone(TimeZone.getDefault());
        //Will print on your default Timezone
        System.out.println(sdf.format(calendar.getTime()));
    }
    public Location getLocationFromAddress(String pAddress) throws
IOException {
        String[] splitArray = pAddress.split("-");
        String address = splitArray[1];
        Log.e("address", address);
        Location pLocation = null;
        Geocoder pGeocoder = new Geocoder(this);
        if (pGeocoder.isPresent())  {
            if (address != "") {
              List<Address> list = pGeocoder.getFromLocationName(address, 1);
               Address address1 = list.get(0);
               double lat = address1.getLatitude();
               double lng = address1.getLongitude();

               Log.e("lat", String.valueOf(lat));
               Log.e("lng", String.valueOf(lng));

               pLocation = new Location("destination");
```

```
                    pLocation.setLatitude(lat);
                    pLocation.setLongitude(lng);
                }
            }
            return (pLocation);
        }
    }
```

6.3. Code to find and store the coordinates of the location using GPS

```
import android.Manifest;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.Service;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.location.Location;
import  android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.os.IBinder;
import android.provider.Settings;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.util.Log;

public class ACGPSTracker extends Service implements LocationListener
{
   private final Context mContext;

   // flag for GPS status
   boolean canGetLocation = false;
```

```
    Location location; // location
    double latitude; // latitude
    double longitude; // longitude

    // The minimum distance to change Updates in meters
    private static final long MIN_DISTANCE_CHANGE_FOR_UPDATES = 10; // 10
meters

    // The minimum time between updates in milliseconds
    private static final long MIN_TIME_BW_UPDATES = 1000 * 60 * 1; // 1 minute

    // Declaring a Location Manager
    protected LocationManager locationManager;

    public ACGPSTracker(Context context)
    {
        this.mContext = context;
        getLocation();
    }

    public Location getLocation()
    {
        Log.e("GPSTracker", "getLocation");

        try {
            locationManager                =                (LocationManager)
mContext.getSystemService(LOCATION_SERVICE);

            // getting GPS status
            boolean                    isGPSEnabled                =
locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER);

            // getting network status
```

```
        boolean isNetworkEnabled = locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER);

        if (!isGPSEnabled && !isNetworkEnabled)
        {
           // no network provider is enabled
        }
        else
           {

           this.canGetLocation = true;
           // First get location from Network Provider
           if (isNetworkEnabled)
           {
              if (ContextCompat.checkSelfPermission(mContext, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED &&
                    ActivityCompat.checkSelfPermission(mContext, Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED)
              {
                 ActivityCompat.requestPermissions((Activity) mContext, new String[]{android.Manifest.permission.ACCESS_FINE_LOCATION, android.Manifest.permission.ACCESS_COARSE_LOCATION}, 101);
              }


locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, MIN_TIME_BW_UPDATES, MIN_DISTANCE_CHANGE_FOR_UPDATES, this);

              Log.d("Network", "Network");
              if (locationManager != null)
              {
```

```
                location =
locationManager.getLastKnownLocation(LocationManager.NETWORK_PROVIDE
R);

                if (location != null)
                {
                   Log.e("ACGPstracker", "isNetworkEnabled");
                   latitude = location.getLatitude();
                   longitude = location.getLongitude();
                }
                else
                {
                   latitude = 0.0;
                   longitude = 0.0;
                }
             }
          }

          // if GPS Enabled get lat/long using GPS Services
          if (isGPSEnabled)
          {
             if (location == null)
             {
                if (ContextCompat.checkSelfPermission(mContext,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
                      ActivityCompat.checkSelfPermission(mContext,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED)
                {
                   ActivityCompat.requestPermissions((Activity) mContext, new
String[]{android.Manifest.permission.ACCESS_FINE_LOCATION,
android.Manifest.permission.ACCESS_COARSE_LOCATION}, 101);
                }
```

```
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
MIN_TIME_BW_UPDATES, MIN_DISTANCE_CHANGE_FOR_UPDATES, this);

                Log.e("GPS Enabled", "GPS Enabled");

                if (locationManager != null)
                {
                   location                                                    =
locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);

                   if (location != null)
                   {
                      Log.e("ACGPstracker", "GPS Enabled");
                      latitude = location.getLatitude();
                      longitude = location.getLongitude();
                   }
                }
             }
          }
       }
       catch (Exception e)
       {
          e.printStackTrace();
       }

       return location;
    }

    /**
     * Stop using GPS listener
     * Calling this function will stop using GPS in your app
```

```
 */

public void stopUsingGPS() {
    if (locationManager != null) {
        locationManager.removeUpdates(ACGPSTracker.this);
    }
}

public double getLatitude()
{
    if (location != null)
    {
        latitude = location.getLatitude();
    }

    return latitude;
}

public double getLongitude()
{
    if (location != null)
    {
        longitude = location.getLongitude();
    }

    // return longitude
    return longitude;
}

public boolean canGetLocation()
{
    return this.canGetLocation;
}
```

```
    @Override
    public void onLocationChanged(Location location) {
    }

    @Override
    public void onProviderDisabled(String provider) {
    }

    @Override
    public void onProviderEnabled(String provider) {
    }

    @Override
    public void onStatusChanged(String provider, int status, Bundle extras) {
    }

    @Override
    public IBinder onBind(Intent arg0) {
        return null;
    }
}
```

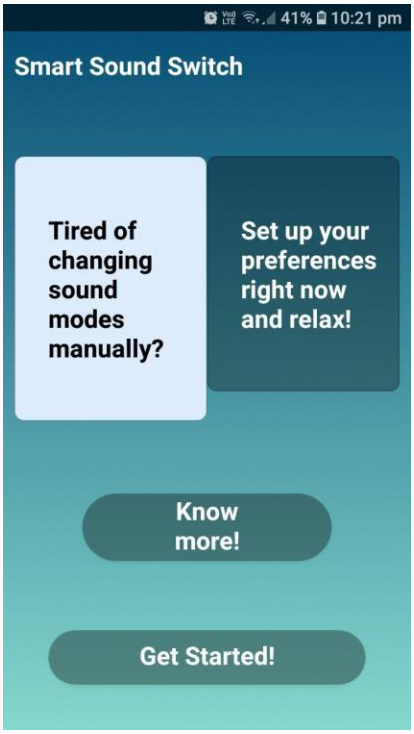
6.4. Code to store and retrieve the timings for a particular event

```
import android.app.TimePickerDialog;
import android.content.Context;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.TimePicker;

import java.util.Calendar;

public class ACSetTime implements /*View.OnFocusChangeListener,*/
TimePickerDialog.OnTimeSetListener, View.OnClickListener {
    private TextView editText;
```

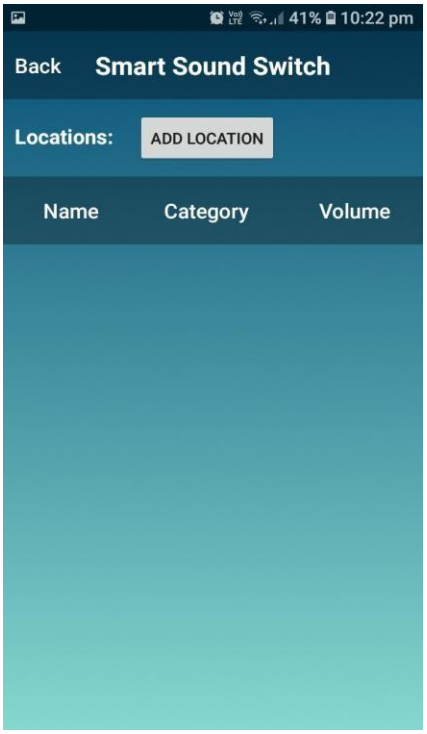
```
private Calendar myCalendar;
private Context ctx;

public ACSetTime(TextView editText, Context ctx)
{
   this.editText = editText;
   this.editText.setOnClickListener(this);
   //this.editText.setOnFocusChangeListener(this);
   this.myCalendar = Calendar.getInstance();
   this.ctx = ctx;
}

@Override
public void onTimeSet(TimePicker view, int hourOfDay, int minute)
{
   // TODO Auto-generated method stub
   //this.editText.setText( hourOfDay + ":" + minute);
   showTime(hourOfDay, minute);
}

public void showTime(int hour, int min)
{
   String format = "";

   if (hour == 0)
   {
      hour += 12;
      format = "AM";
   }
   else if (hour == 12)
   {
      format = "PM";
   }
   else if (hour > 12)
```

```
        {
            hour -= 12;
            format = "PM";
        }
        else
        {
            format = "AM";
        }

        StringBuilder pStringBuilder = new StringBuilder();
        pStringBuilder.append(hour);
        pStringBuilder.append(":");
        pStringBuilder.append(min);
        pStringBuilder.append(" ");
        pStringBuilder.append(format);

        this.editText.setText(pStringBuilder);
    }

    @Override
    public void onClick(View v)
    {
        int hour = myCalendar.get(Calendar.HOUR_OF_DAY);
        int minute = myCalendar.get(Calendar.MINUTE);
        TimePickerDialog pTimePickerDialog = new TimePickerDialog(ctx, this, hour,
minute, false);
        pTimePickerDialog.show();
    }
}
```

7. OUTPUT

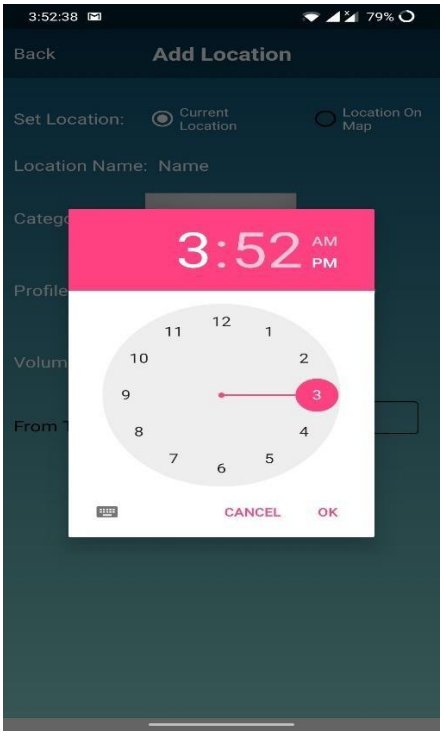


7.1 Home Page

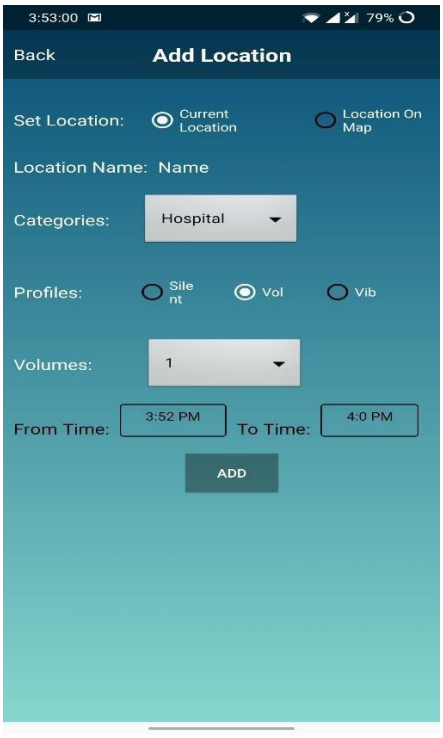


7.2 Events Page

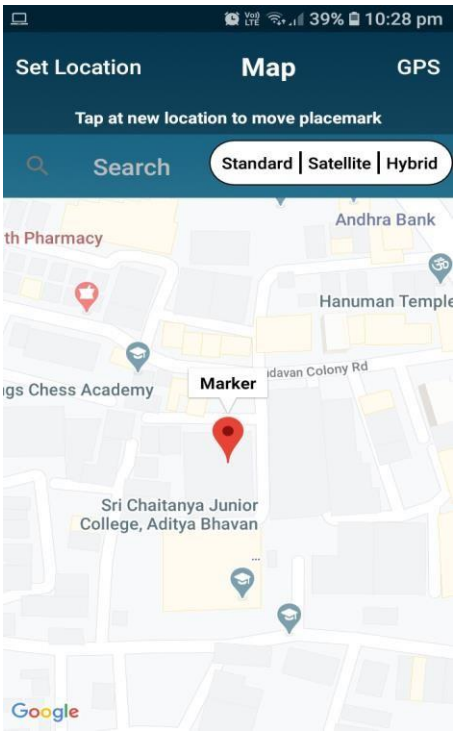
7.3 Adding Categories and Profile



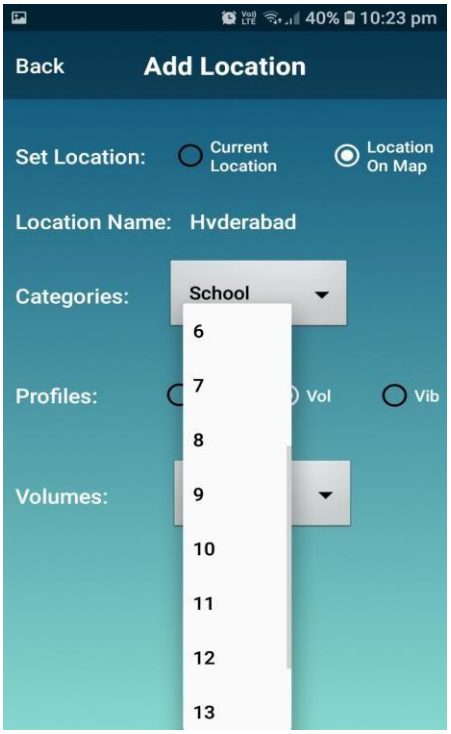
7.4 Adding Timings



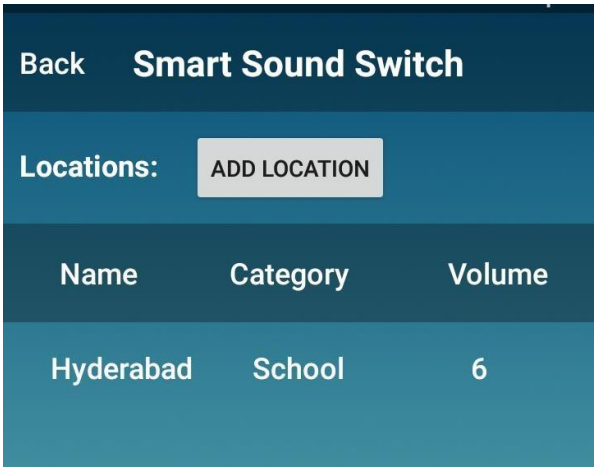
7.5 Adding Event



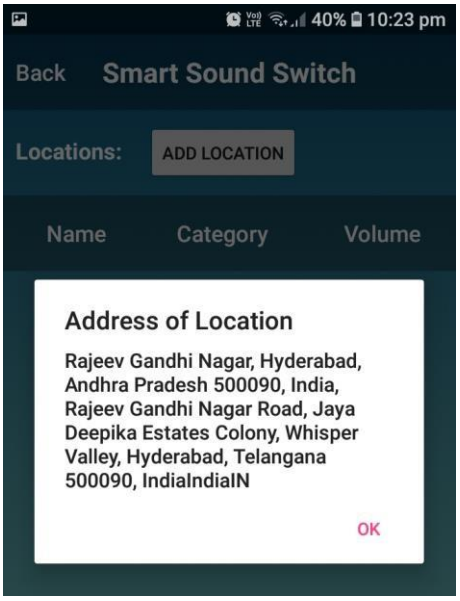
7.6 Adding Location



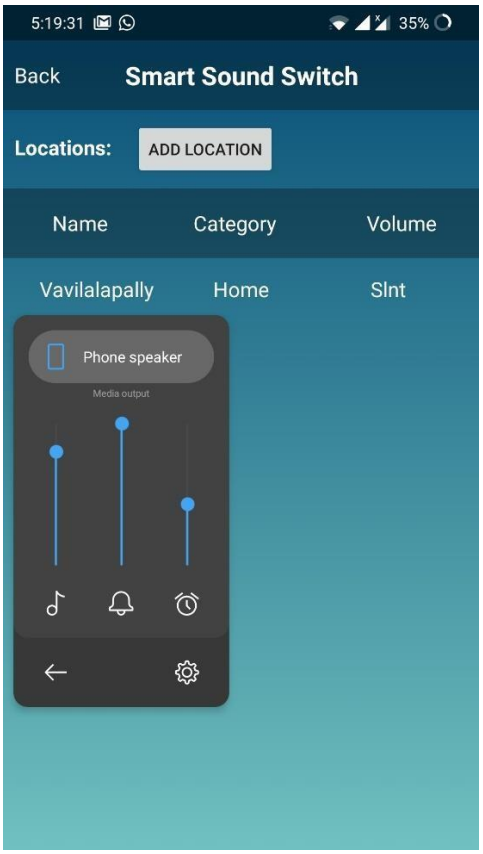
7.7 Profile Mode Settings



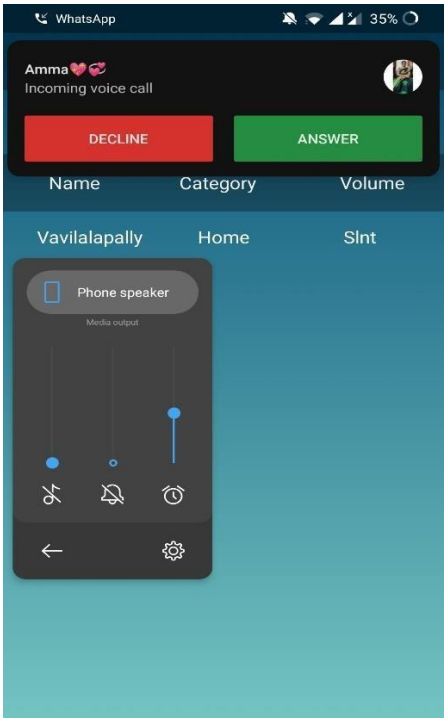
7.8 Added Events



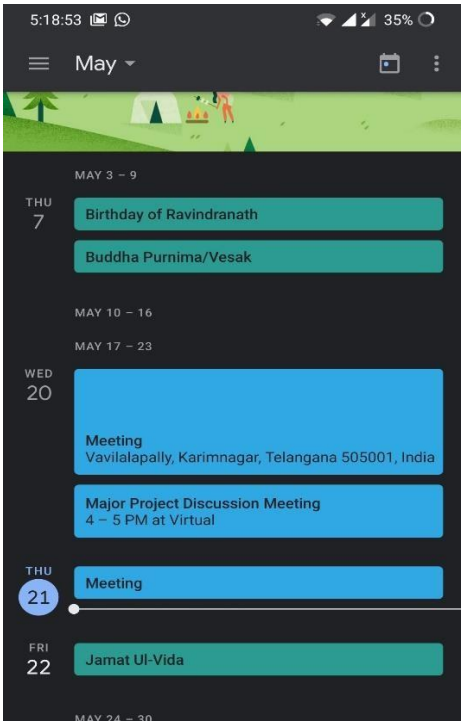
7.9 Added Location



8.0 Sound Profiles Before Call



8.1 Sound Profiles During Call



8.2 Calender Events

8. CONCLUSION AND FUTURE SCOPE

8.1 CONCLUSION

This project includes the entire study of Location based automatic profile changer. It gives the information of location service in a mobile, where each node needs to maintain its location information by frequently updating its location information within its neighboring region, which is called neighborhood update (NU), and occasionally updating its location information to certain distributed location server in the network, which is called location server update (LSU). In this ways it provided a systematic way of profile changing according to the GPS coordinates and also reminds us about it and makes the use of mobile to the user more convenient. Location Based Automatic Sound Profile Switching Application in Android Mobiles is a next level of Location Aware Intelligent Software which reduces human intervention for simple task such as sound profile switching. Android Smart Phone becomes much smarter by this application.

8.2 FUTURE SCOPE

In a nutshell, it can be summarized that the future scope of our project circles around the following points:

- Instead of profile switching one can design this application for call divert also, so whenever he enters into the Silent Zone his all calls will be diverted on some another number specified by him.
- User-defined accuracy settings for user-defined locations.

These are the enhancements which can be done to increase the better usage and scope of the application.

9. REFERENCES

- Bharti Ahuja, Mayuri Deshmukh, Ruchika Borhade and Pooja Nikam, "Location Based Automatic Profile Changer and Mobiminder ", IJETAE Exploring research and innovation, April 2013
- Rohit Madhukar Chaskar, "LOCATION BASED AUTOMATIC SOUND PROFILE SWITCHING APPLICATION IN ANDROID MOBILES", International Journal of Computer & Communication Technology ISSN, 2017
- Vikram Kumar, Eniyamaran and Evinston Wilson Shalom, "AUTO SILENT SYSTEM USING LOCATIONS SERVICES IN ANDROID MOBILES", IJARCSEIT, isrjournal, April 2017
- Google http://www.android.com/
- P. Enge, and P. Misra, “Special Issue on GPS:The Global positioning System”, Proc. of the IEEE, pp. 3-172, Jan 1999
- Ch. Radhika Rani, A. Praveen Kumar, D. Adarsh, K. Krishna Mohan, K.V.Kiran, “Location Based Services in Android”, International Jour nal of Advances in Engineering & Technology, ISSN: 2231-1963, Mar 2012
- http://www.vogella.de/articles/Android/article.html#overvie w_android
- http://developer.android.com/resources/samples/Home/inde x.html