

**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**JnanaSangama, Belgaum-590014**



**A Mobile Application Development Mini Project**

**Report On**

**“MEETING SCHEDULE”**

**Submitted in Partial fulfilment of the Requirements for the VI semester of the Degree of**

**Bachelor of Engineering**

**In**

**Computer Science & Engineering**

**By**

**SURABHI G R(1CE18CS084)**

**CHETANA NATH (1CE18CS061)**

**Under the Guidance of**

**Mrs. SHASHIKALA SANNU**

**Professor, Dept. Of CSE**



**CITY ENGINEERING COLLEGE**

**Doddakallasandra, Kanakapura Road,**

**Bengaluru-560061**

# CITY ENGINEERING COLLEGE

Doddakallasandra, Kanakapura Road, Bengaluru-560061

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



### CERTIFICATE

Certificate that the **MOBILE APPLICATION DEVELOPEMNT** Mini Project work entitled **“MEETING SCHEDULE”** has been carried out by **SURABHI GR (1CE18CS084)** and **CHETANA NATH (1CE18CS061)**, bonafide students of City Engineering College in partial fulfilment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year **2020-2021**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The **MOILE APPLICATION DEVELOPEMNT** Mini Project Report has been approved as it satisfies the academic requirements in respect of project workprescribed for the said Degree.

**Mrs. Shashikala Sannu**  
Prof, Dept. Of CSE

**Mr. B Vivekavardhana Reddy**  
Head, Dept. Of CSE

**Dr. Thippeswamy H N**  
Principal

External Viva

Name of the examiners

Signature with date

1.

2.

## **ABSTRACT**

In order to solve the problem of complex meeting schedules on the current market, a new app of simple, convenient, less required memory as well as user-friendly is developed. Based on the Android technology, using the Java language and Eclipse programming tools lead to design and coding.

The new design mainly realizes five core functions including date, time, meeting agenda, location and the meeting held with. This app has merits of high performance, simple operation, and run independently on the Android mobile devices. At the same time, the app will automatically files in mobile phones. This app is more specific and MORE PRICISED.

## ACKNOWLEDGEMENT

While presenting this COMPUTER GRAPHICS project on “**MEETING SCHEDLUE**”, we feel that it is our duty to acknowledge the help rendered to us by various persons.

Firstly, we thank God for showering his blessings on us. We are grateful to our institution City Engineering College for providing us a congenial atmosphere to carry out the project successfully.

We would like to express our heartfelt gratitude to, **Dr. Thippeswamy H N**, Principal, CEC, Bangalore, for extending his support.

We would also like to express our heartfelt gratitude to **Prof. Vivekavardhana Reddy**, HOD, Computer Science and Engineering whose guidance and support was truly invaluable.

We are very grateful to our guide, **Mrs. Shashikala Sannu** Prof., Department of Computer Science, for their able guidance and valuable advice at every stage of our project which helped me in the successful completion of our project.

We would also have indebted to our parent and friends for their continued moral and materialsupport throughout the course of project and helping me in finalize the presentation.

Our hearty thanks to all those have contributed bits, bytes and words to accomplish this project.

SURABHI G R (1CE18CS084)

CHRTANA NATH (1CE18CS061)

# TABLE OF CONTENTS

SL NO.	CHAPTERS	PAGE NO.
1.	<b>INTRODUCTION</b>	
	1.1 Introduction of developing environment of Android	1
	1.2 Build developing environment of Android.	1
	1.3 The required software of the developing environment.	2
	1.4 The design principle of android application	2
	1.5 Function and structure design of Android system	3
2.	<b>Requirement analysis of system</b>	
	2.1 The feasibility analysis.	4
	2.2 Hardware Requirements to Run the APP.	5
	2.3 Other Requirements.	5
3.	<b>System Design</b>	
	3.1 Saving Data using SQL Lite.	6
	3.2 Header Files	7
	3.3 Introduction of App Starting module in the project	10
	3.4 Introduction of engineering program structure	11
	3.5 Part of the function design.	12
	3.6 The extension mobile audio files access function to mobile	12
	3.7 Data Storage	12
4.	<b>Flow Chart</b>	13
5.	<b>Source Code</b>	14
6.	<b>Snapshots</b>	28
	<b>Conclusion</b>	30
	<b>Future Enhancement</b>	30
	<b>Bibliography</b>	31

## CHAPTER 1

### INTRODUCTION

Android is open source code mobile phone operating system that comes out by Google in November 2007. Its appearance has broken the traditional closed mobile phone operating system. Anyone can modify the mobile phone operating system as well as function according to personal preference, which is also the most attractive merit of Android. Meeting schedule in this article is application software based on Google Android.

Android's application on mobile terminals also completely broke the traditional understanding of the mobile terminals. Therefore, many kinds of mobile phone database is also developed. However, a lot of database devote to fancy appearance and function, while caused resources wasting to the user's mobile phone, such as large required memory and CPU, which brings a lot of inconvenience as multiple programs running at the same time. For the most ordinary users, many functions are useless.

The purpose of this article is to develop a database which can record and alert the user according to the schedule. To browse and query the storage space as well as operation of adding, deleting, and modifying can be realized.

Meeting schedule is based on Android application is popular in the market at the present. The completing development of Android operating system gives developers a nice platform, which can learn the popular computer technology combining with learned knowledge, and master the latest knowledge, enrich oneself, and enjoy entertainment.

#### **1.1 Introduction of developing environment of Android**

This chapter is mainly to study and introduce the needed platform for Android Database, and introduction of the needed configuration environment.

#### **1.2 Build developing environment of Android.**

The applications of Android need to run based on Android environment. The following is the configuration requirement and installation steps of Android development environment.

### **1.3 The required software of the developing environment**

Operation system: Windows XP / Linux / Windows 7

Software Android SDK(Software Development Kit)、ADT(Android Development Tool)

IDE environment Eclipse IDE + ADT Eclipse3 or higher

JDK: Java Runtime Environment virtual machine-Java Development Kit(JDK) Installation steps of the developing environment

Step 1: install the Java virtual machine JDK version - 6

Step 2: install Eclipse3-5 tools; download address: <http://www-eclipse-org/downloads/>

Step 3: install the Android SDK: first download the Android SDK

Download address: <http://developer-android-com/sdk/index-html>

Step 4:

Install Android ADT plug-in, run Eclipse and select help - > install new software and select add.

Input SDK tools path in the SDK location: D: \ android \ software \ android SDK – Windows and click OK. The Android environment is set up successfully.

### **1.4 The design principle of android application.**

Twice the result with half the effort will get if an overall study of the principles done before the design and follow them in the operation. The principle of software design mainly includes the following points:

#### **(1) Reliability**

The reliability of the software design must be determined. The reliability of the software system refers to the ability to avoid fault occurred in the process of system running, as well as the ability to remedy troubles once the fault occurs.

#### **(2) Reusability**

Look for commonness of similar codes, and come out new method abstractly and reasonably. Pay attention to the generic design.

**(3) Understandability**

The understandability of software not only require clear and readable document, but the simplified structure of software itself, which requires the designer possess keen insight and creativity, and know well about the design objects.

**(4) Simple program**

To keep the program simple and clear, good programmers can use simple program to solve complex problems.

**(5) Testability**

Testability means that the created system has a proper data collection to conduct a comprehensive test of the entire system.

**(6) The Open-Closed Principle**

Module is extensible but cannot be modified. That is to say, extension is open to the existing code in order to adapt to the new requirements. While modify is closed to the categories. Once the design is completed, the categories cannot be modified.

**1.5 Function and structure design of Android system.**

This system adopts the modularized program design, and system function is correspondingly divided into function modules, the main modules include:

- (1) UI function module design of mobile terminal: the medicine name, time of the day, time, for, Prescribed by.
- (2) Backstage function module design of mobile terminal: the specific function, music file data storage function and other function are implemented.



**CHAPTER 2****REQUIREMENT ANALYSIS OF SYSTEM****2.1 The feasibility analysis: -**

This section verified that it is feasible to add database on the Android system from the aspects of economic, technical and social feasibility.

**Economic feasibility:**

To design Android mobile phone database as long as a computer has the Android development and the application development of Android is free. In addition, mobile phone music player is basic needs for public. The information that which functions are necessary form all the consumers, which functions are needed for some people, and which features are seldom to use is easy to understand. And a lot of research is eliminated, thus saved the spending. Therefore, the whole process of development doesn't need to spend any money that is economic feasibility.

**Technical feasibility:**

To design a database which meets the basic requirements, a deep understand of JAVA language, the Eclipse development tools, SQLite databases, the Android system architecture, application of framework and other technical knowledge are needed. (framework is the core of the application, and rules that all the programmers participating in the development must abide by). Based on the related technology information and resources for Android on the market, and equipped with technical personnel of technology and the spirit of willing to learn, the technology is feasible.

**Social feasibility:**

With the rapid development of the mobile phone market, all kinds of audio and video resources are widely circulated on the Internet. These resources seem ordinary, but have gradually become an indispensable part of people life, which derived the development of all kinds of mobile phone player. But a lot of players devoted to fancy appearance, strong function causing a lot of wasted resources to the user's mobile phone and bringing a lot of inconvenience to the

user as multitasking operation is needed. Some functions are useless to ordinary people. Powerful player is a good thing, but a lot of functions are actually useless for most users

## **2.2 Hardware Requirements to Run the APP**

**This APP meets minimum hardware Requirements:**

- ☐ 1 GHz Processor
- ☐ 512MB of RAM
- ☐ 50MB of Internal Storage

## **2.3 OTHER REQUIREMENTS**

- Maintain ability:  
The design will be updated based on any changes, which are done during Coding stage to maintain proper trace ability.
- Availability:  
Available for minimum API level 22 (Android Lollipop 5.1)

**CHAPTER 3****SYSTEM DESIGN**

In this chapter, design steps and the results of functional modules in the system are given in details.

**3.1 Saving Data using SQLite:**

SQLite is a opensource SQL database that stores data to a text file on a device. Android comes in with built in SQLite database implementation.

SQLite supports all the relational database features. In order to access this database, you don't need to establish any kind of connections for it like JDBC,ODBC e.t.c

**Database - Package**

The main package is android.database.sqlite that contains the classes to manage your own databases

**Database - Creation**

In order to create a database you just need to call this method openOrCreateDatabase with your database name and mode as a parameter. It returns an instance of SQLite database which you have to receive in your own object

**Database - Fetching**

We can retrieve anything from database using an object of the Cursor class. We will call a method of this class called rawQuery and it will return a resultset with the cursor pointing to the table.

**Database - Helper class**

For managing all the operations related to the database , an helper class has been given and is called SQLiteOpenHelper. It automatically manages the creation and update of the database.

### 3.2 Header Files: -

**import android.Manifest;** The Android Manifest is an XML file which contains important metadata about the Android app. This includes the package name, activity names, main activity (the entry point to the app), Android version support, hardware features support, permissions, and other configurations.

**import android.app.SearchManager;** This class provides access to the system search services. In practice, you won't interact with this class directly, as search services are provided through methods in Activity and the ACTION\_SEARCH Intent

**import android.content.Context;** Context provides the connection to the Android system and the resources of the project. It is the interface to global information about the application environment. The Context also provides access to Android Services, e.g. the Location Service. Activities and Services extend the Context class.

**import android.content.DialogInterface;** Interface used to allow the creator of a dialog to run some code when an item on the dialog is clicked.

**import android.content.pm.PackageManager;** Class for retrieving various kinds of information related to the application packages that are currently installed on the device.

**import android.os.Handler;** A Handler allows you to send and process Message and Runnable objects associated with a thread's MessageQueue . Each Handler instance is associated with a single thread and that thread's message queue. It will deliver messages and runnables to that Looper's message queue and execute them on that Looper's thread.

**import android.support.annotation.NonNull;** The @Nullable annotation can be used to indicate that a given parameter or return value can be null. Similarly, the @NonNull annotation can be used to indicate that a given parameter (or return value) can not be null.

**import android.support.design.widget.FloatingActionButton;** Floating action button is used for a special type of promoted action, it animates onto the screen as an expanding piece of material, by default. The icon within it may be animated, also FAB may move differently than other UI elements because of their relative importance. A floating action button represents the primary action in an application which can simply trigger an action or navigate somewhere.

**import android.support.design.widget.NavigationView;** Represents a standard navigation menu for application. The menu contents can be populated by a menu resource file. NavigationView is typically placed inside a DrawerLayout .

**import android.support.design.widget.Snackbar;** Snackbar in android is a new widget introduced with the Material Design library as a replacement of a Toast. Android Snackbar is light-weight widget and they are used to show messages in the bottom of the application with swiping enabled. Snackbar android widget may contain an optional action button.

**import android.support.design.widget.TabLayout;** TabLayout is used to implement horizontal tabs. TabLayout is released by Android after the deprecation of ActionBar. Tabs are created using newTab() method of TabLayout class. The title and icon of Tabs are set through setText(int) and setIcon(int) methods of TabListener interface respectively.

**import android.support.v4.app.ActivityCompat;** ActivityCompat. A helper for accessing features in Activity in a backwards compatible fashion. Construct this by using getActivityCompat(Activity) .

**import android.support.v4.content.ContextCompat;** ContextCompat class is used when you would like to retrieve resources, such as drawable or color without bother about theme. It provide uniform interface to access resources and provides backward compatibility. Common use case could be get color or drawable, etc.

**import android.support.v4.view.ViewPager;** The ViewPager is the widget that allows the user to swipe left or right to see an entirely new screen. In a sense, it's just a nicer way to show the user multiple tabs. It also has the ability to dynamically add and remove pages (or tabs) at any time.

**import android.support.v4.widget.DrawerLayout;** DrawerLayout acts as a top-level container for window content that allows for interactive "drawer" views to be pulled out from one or both vertical edges of the window.

**import android.support.v7.app.ActionBar;** A primary toolbar within the activity that may display the activity title, application-level navigation affordances, and other interactive items.

**import android.support.v7.app.AlertDialog;** A subclass of Dialog that can display one, two or three buttons. If you only want to display a String in this dialog box, use the setMessage() method.

**import android.support.v7.app.AppCompatActivity;** This is a compatibility library that back ports some features of recent versions of Android to older devices.

**import android.os.Bundle;** Android Bundle is used to pass data between activities. The values that are to be passed are mapped to String keys which are later used in the next activity to retrieve the values.

**import android.view.Gravity;** android:gravity is an attribute that sets the gravity of the content of the view its used on. The android:gravity specifies how an object should position its content on both X and Y axis. The possible values of android:gravity are top, bottom, left, right, center, center\_vertical, center\_horizontal etc.

**import android.view.Menu;** Android Option Menus are the primary menus of android. They can be used for settings, search, delete item etc. Here, we are inflating the menu by calling the inflate() method of MenuInflater class. To perform event handling on menu items, you need to override onOptionsItemSelected() method of Activity class.

**import android.support.v7.widget.Toolbar;** A Toolbar is a generalization of action bars for use within application layouts. While an action bar is traditionally part of an Activity's opaque window decor controlled by the framework, a Toolbar may be placed at any arbitrary level of nesting within a view hierarchy. An application may choose to designate a Toolbar as the action bar for an Activity using the setSupportActionBar() method.

**import android.widget.ImageButton;** This widget is merely a simple wrapper around a long-press handler. Displays a button with an image (instead of text) that can be pressed or clicked by the user. By default, an ImageButton looks like a regular Button, with the standard button background that changes color during different button states.

**import android.widget.SearchView;** A widget that provides a user interface for the user to enter a search query and submit a request to a search provider. Shows a list of query suggestions or results, if available, and allows the user to pick a suggestion or result to launch into.

**import android.widget.SeekBar;** A SeekBar is an extension of ProgressBar that adds a draggable thumb. The user can touch the thumb and drag left or right to set the current progress level or use the arrow keys. Placing focusable widgets to the left or right of a SeekBar is discouraged.

**import android.widget.TextView;** A TextView displays text to the user and optionally allows them to edit it. A TextView is a complete text editor, however the basic class is configured to not allow editing.

**import android.widget.Toast;** A toast is a view containing a quick little message for the user. The toast class helps you create and show those. When the view is shown to the user, appears as a floating view over the application.

**import java.util.ArrayList;** The java.util.ArrayList class provides resizable-array and implements the List interface. Following are the important points about ArrayList – It implements all optional list operations and it also permits all elements, includes null.

### **3.3 Introduction of AppStarting module in the project:**

Any AppStarting needs AndroidManifest. XML file to start. And any new project content will automatically generate an AndroidManifest. XML file. Configuration files are the core of the whole program, which contains the Android SDK version, and the default Activity in program running.

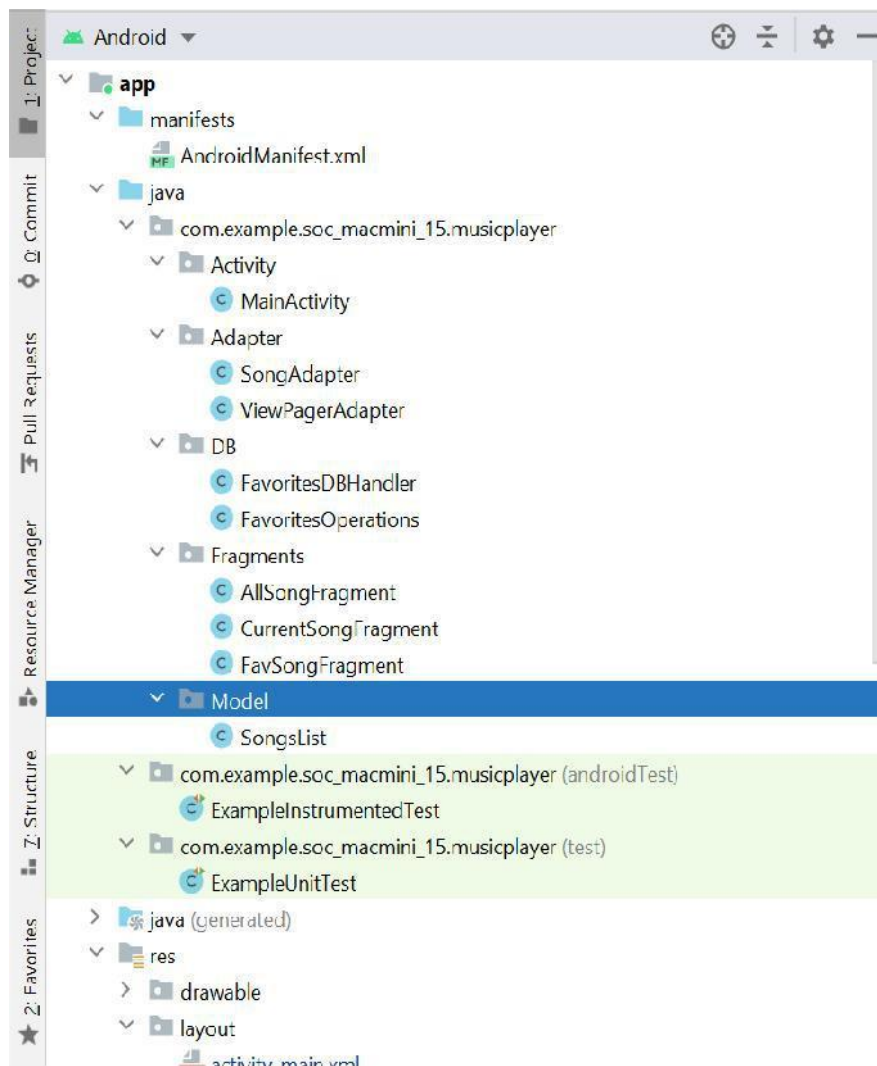
The systems will automatically be looking for a logo in AndroidManifest to react the corresponding operation when any component of the program triggers events.

To define the system, the first thing is launching the Activity: Android Activity. There are properties such as action and category in < intent - filter >. Most of these are the default values of the system. Setting the action and category realize the switch between different Activities.

When any components of the program is about to use, declaration must be in the Android Manifest. Xml files.

### 3.4 Introduction of engineering program structure

The basic structure content of Android project includes: the SRC (source code), gen (constant that Android system automatically generates), res (resource file), and the layout of file in the main storage program interface, which is shown as in fig 3.1.





### **3.5 Part of the function design:**

The main play interface design:

Convenience and practical should be fully considered in the design of the main interface. Every Android interface is a visual interface, which has its unique layout configuration files. We can configure various layout and resources files according to the requirements, such as images, text and color reference, which can form different visual interface and glaring effect. Interface design of adding songs. There are no corresponding songs for the first-time login entering the program; users need to add songs to play. Therefore, you need to enter the adding songs' interface. The empty playlist needs to add songs which can choose from the SD card to add.

Function design of play and Next/Move Previous music When need to use the player to play appropriate music, click the play button to realize the function. When need to use the player to switch to the previous song, click on "Move Previous music "button to realize the function.

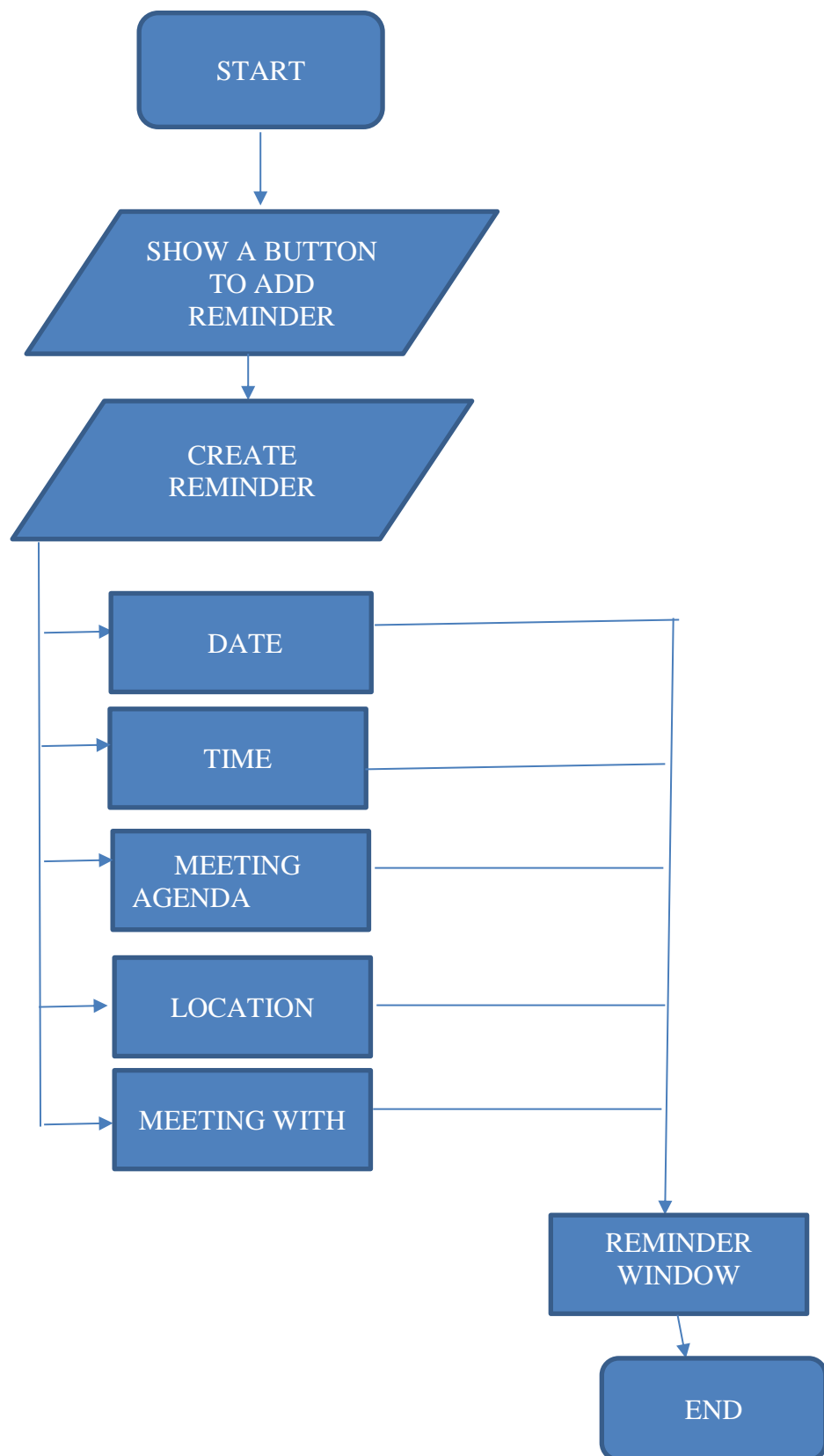
When need to use the player to play the next song, click on "the next music" button to realize the function.

### **3.6 The extension mobile audio files access functions of mobile phones**

The application implements the function of the file browser. As a file browser, it must have the function of browsing. When the program is run to browsing interface, all contents of the files and icons will appear. We can see all the files from the file browser, which also can be edited. This program is designed for adding songs of the player, so browsing function is limited to the media file and content browsing containing media files.

### **3.7 Data storage**

When players run normally, because of the switch among interface, in order to avoid the data lost, we need to store some data for temporary or permanent storage. As a kind of mobile phone operating system, Android provides the following ways for data storage: Preference (configuration), File (documents), SQLite data and network. Application component between each other is independent In Android, and data cannot be shared. In order to realize data sharing, Android provides Content Provider components to realize the sharing of data among applications.

**FLOW CHART**

**CHAPTER 5****SOURCE CODE****activity\_main.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:background="@drawable/background"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="343dp"
        android:layout_height="82dp"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentBottom="true"
        android:layout_marginLeft="35dp"
        android:layout_marginTop="19dp"
        android:layout_marginRight="32dp"
        android:layout_marginBottom="630dp"
        android:fontFamily="@font/almendra"
        android:gravity="center"
        android:text="Meet Them"
        android:textColor="#FFFFFF"
        android:textSize="60sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent">
```

```
app:layout_constraintTop_toTopOf="parent" />
```

```
<TextView
```

```
    android:id="@+id/textView3"
    android:layout_width="210dp"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentBottom="true"
    android:layout_marginLeft="103dp"
    android:layout_marginBottom="615dp"
    android:fontFamily="@font/acme"
    android:text="take care of ur schedule"
    android:textColor="#90CAF9"
    android:textSize="20sp"
    android:textStyle="italic" />
```

```
<Button
```

```
    android:id="@+id/bt1"
    android:layout_width="277dp"
    android:layout_height="64dp"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="false"
    android:layout_alignParentBottom="true"
    android:layout_marginLeft="67dp"
    android:layout_marginBottom="30dp"
    android:background="@drawable/button"
    android:fontFamily="@font/amaranth"
    android:shadowColor="#000000"
    android:text="Create a schedule"
    android:textSize="15sp"
    app:backgroundTint="#4C97BF" />
```

```
</RelativeLayout>
```

**MainActivity.java**

```
package com.example.meetingschedule;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.Button;
import android.content.Intent;
import android.view.View;

public class MainActivity extends AppCompatActivity {
    public Button button;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        button = (Button) findViewById(R.id.bt1);

        button.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                Intent intent= new Intent(MainActivity.this,MainActivity2.class);
                startActivity(intent);
            }
        });
    }
}
```

**activity\_main2.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
```

```
android:background="@drawable/background"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">
```

```
<TextView
```

```
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentBottom="true"
    android:layout_marginLeft="85dp"
    android:layout_marginBottom="662dp"
    android:fontFamily="@font/acme"
    android:text="Create a schedule"
    android:textColor="#90CAF9"
    android:textSize="35sp" />
```

```
<EditText
```

```
    android:id="@+id/t1"
    android:layout_width="361dp"
    android:layout_height="49dp"
    android:layout_alignParentStart="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentEnd="true"
    android:layout_alignParentRight="true"
    android:layout_marginStart="24dp"
    android:layout_marginLeft="24dp"
    android:layout_marginTop="103dp"
    android:layout_marginEnd="25dp"
    android:layout_marginRight="25dp"
    android:background="@drawable/input_field"
    android:ems="10"
    android:fontFamily="@font/acme"
```

```
android:hint="Date"
android:inputType="date"
android:shadowColor="#86000000"
android:shadowRadius="0"
android:textColor="#000000"
android:textColorHint="@color/black" />
```

<EditText

```
android:id="@+id/t2"
android:layout_width="361dp"
android:layout_height="49dp"
android:layout_below="@+id/t1"
android:layout_alignParentStart="true"
android:layout_alignParentLeft="true"
android:layout_alignParentEnd="true"
android:layout_alignParentRight="true"
android:layout_marginStart="23dp"
android:layout_marginLeft="23dp"
android:layout_marginTop="28dp"
android:layout_marginEnd="25dp"
android:layout_marginRight="25dp"
android:background="@drawable/input_field"
android:ems="10"
android:fontFamily="@font/acme"
android:hint="Time"
android:inputType="time"
android:shadowColor="#000000"
android:shadowRadius="0"
android:textColor="#000000"
android:textColorHint="@color/black" />
```

<EditText

```
android:id="@+id/t3"
android:layout_width="361dp"
```

```
android:layout_height="49dp"
android:layout_below="@+id/t2"
android:layout_alignParentStart="true"
android:layout_alignParentLeft="true"
android:layout_alignParentEnd="true"
android:layout_alignParentRight="true"
android:layout_marginStart="24dp"
android:layout_marginLeft="24dp"
android:layout_marginTop="27dp"
android:layout_marginEnd="25dp"
android:layout_marginRight="25dp"
android:background="@drawable/input_field"
android:ems="10"
android:fontFamily="@font/acme"
android:hint="Meeting Agenda"
android:inputType="textPersonName"
android:shadowColor="#000000"
android:shadowRadius="0"
android:textColor="#000000"
android:textColorHint="@color/black" />
```

<EditText

```
android:id="@+id/t4"
android:layout_width="361dp"
android:layout_height="49dp"
android:layout_below="@+id/t3"
android:layout_alignParentStart="true"
android:layout_alignParentLeft="true"
android:layout_alignParentEnd="true"
android:layout_alignParentRight="true"
android:layout_marginStart="25dp"
android:layout_marginLeft="25dp"
android:layout_marginTop="26dp"
android:layout_marginEnd="24dp"
```



```
android:layout_marginRight="24dp"
android:background="@drawable/input_field"
android:ems="10"
android:fontFamily="@font/acme"
android:hint="Location"
android:inputType="textPersonName"
android:shadowColor="#000000"
android:shadowRadius="0"
android:textColor="#000000"
android:textColorHint="@color/black" />
```

<EditText

```
android:id="@+id/t5"
android:layout_width="361dp"
android:layout_height="49dp"
android:layout_below="@+id/t4"
android:layout_alignParentStart="true"
android:layout_alignParentLeft="true"
android:layout_alignParentEnd="true"
android:layout_alignParentRight="true"
android:layout_marginStart="23dp"
android:layout_marginLeft="23dp"
android:layout_marginTop="28dp"
android:layout_marginEnd="25dp"
android:layout_marginRight="25dp"
android:background="@drawable/input_field"
android:ems="10"
android:fontFamily="@font/acme"
android:hint="Client Name"
android:inputType="textPersonName"
android:shadowColor="#000000"
android:shadowRadius="0"
android:textColor="#000000"
android:textColorHint="@color/black" />
```

<Button

```
    android:id="@+id/but1"
    android:layout_width="277dp"
    android:layout_height="64dp"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="false"
    android:layout_alignParentBottom="true"
    android:layout_marginLeft="67dp"
    android:layout_marginBottom="45dp"
    android:background="@drawable/button"
    android:fontFamily="@font/amaranth"
    android:shadowColor="#000000"
    android:text="Create a schedule"
    android:textSize="15sp"
    app:backgroundTint="#4C97BF"
    android:onClick="addRecord"/>
```

</RelativeLayout>

## **MainActivity.java**

```
package com.example.meetingschedule;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity2 extends AppCompatActivity
```

```
{
    EditText t1,t2,t3,t4,t5;
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);
        t1=(EditText)findViewById(R.id.t1);
        t2=(EditText)findViewById(R.id.t2);
        t3=(EditText)findViewById(R.id.t3);
        t4=(EditText)findViewById(R.id.t4);
        t5=(EditText)findViewById(R.id.t5);
    }
    public void startdbapp(View view)
    {
        new DbManager(this);
        startActivity(new Intent(this,InsertData.class));
    }
    public void addRecord(View view)
    {
        DbManager db= new DbManager(this);
        String res
=db.addRecord(t1.getText().toString(),t2.getText().toString(),t3.getText().toString(),t3.getTe
xt().toString(),t5.getText().toString());
        Toast.makeText(this,res, Toast.LENGTH_SHORT).show();
        t1.setText("");
        t2.setText("");
        t3.setText("");
        t4.setText("");
        t5.setText("");
    }
}
```

input\_field.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<shape android:shape="rectangle"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <solid android:color="#F2F2F2"/>
    <corners android:radius="11dp"/>
    <padding android:left="10dp"/>
    <stroke android:color="#C9D6FF" android:width="02dp"/>
</shape>
```

## **InsertData.java**

```
package com.example.meetingschedule;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

import java.text.BreakIterator;

public class InsertData extends AppCompatActivity
{
    EditText t1,t2,t3,t4,t5;
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);
        t1=(EditText)findViewById(R.id.t1);
        t2=(EditText)findViewById(R.id.t2);
        t3=(EditText)findViewById(R.id.t3);
        t4=(EditText)findViewById(R.id.t4);
        t5=(EditText)findViewById(R.id.t5);
    }
}
```

```
public void addRecord(View view)
{
    DbManager db= new DbManager(this);
    String res
=db.addRecord(t1.getText().toString(),t2.getText().toString(),t3.getText().toString(),t3.getTe
xt().toString(),t5.getText().toString());
    Toast.makeText(this,res, Toast.LENGTH_SHORT).show();
    t1.setText("");
    t2.setText("");
    t3.setText("");
    t4.setText("");
    t5.setText("");
}
}
```

### **DbManager.java**

```
package com.example.meetingschedule;
import android.content.ContentValues;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

import androidx.annotation.Nullable;

public class DbManager extends SQLiteOpenHelper
{
    private static final String dbname = "medicine.db";
    public DbManager(Context context)
    {
        super(context, dbname, null, 1);
    }

    @Override
```

```
public void onCreate(SQLiteDatabase db)
{
    String qry="create table tabl_meeting(date text, time text, location text, meetingAgenda
text,name text)";
    db.execSQL(qry);
}
```

@Override

```
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
{
    db.execSQL("DROP TABLE IF EXISTS tabl_meeting");
    onCreate(db);
}

public String addRecord(String p1,String p2,String p3,String p4,String p5)
{
    SQLiteDatabase db= this.getWritableDatabase();
    ContentValues cv=new ContentValues();
    cv.put("date",p1);
    cv.put("time",p2);
    cv.put("location",p3);
    cv.put("meetingAgenda",p4);
    cv.put("name",p5);

    long res=db.insert("tabl_meeting",null,cv);
    if(res==-1)
        return "Failed";
    else
        return "MEETING SCHEDULED!";
}
}
```

### **button.xml**

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<shape android:shape="rectangle"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <corners android:radius="25dp"/>
    <gradient
        android:startColor = "#2980B9"
        android:centerColor = "#6DD5FA"
        android:endColor = "#2C5364"
        android:angle = "90"
        android:type = "linear" />
    <size android:width="412dp"/>
    <size android:height="168dp"/>
</shape>
```

### **background.xml**

```
<?xml version = "1.0" encoding = "utf-8" ?>
<shape
    android:shape = "rectangle"
    xmlns:android = "http://schemas.android.com/apk/res/android" >

    <gradient
        android:startColor = "#0F2027"
        android:centerColor = "#203A43"
        android:endColor = "#2C5364"
        android:angle = "270"
        android:type = "linear" />
</shape>
```

### **AndroidManifest.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.meetingschedule">

    <application
```

```
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.MeetingSchedule">
    <activity android:name=".MainActivity2"></activity>
    <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>

    <meta-data
        android:name="preloaded_fonts"
        android:resource="@array/preloaded_fonts" />
</application>

</manifest>
```



## SNAPSHOTS

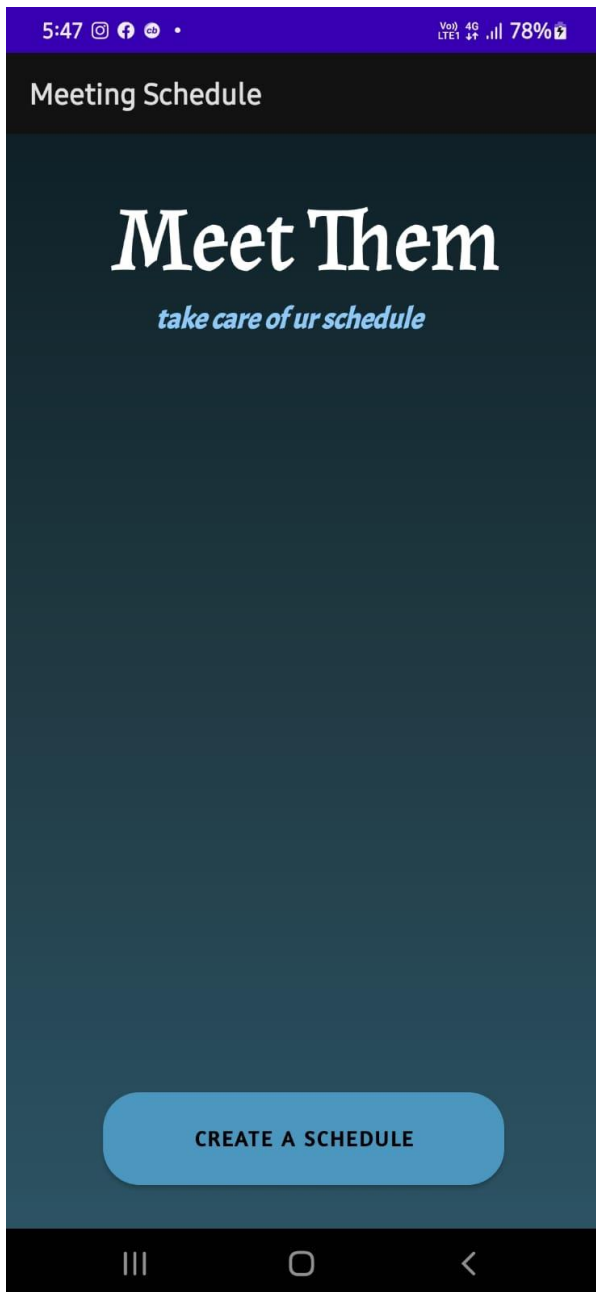


Fig 1: Starting Page

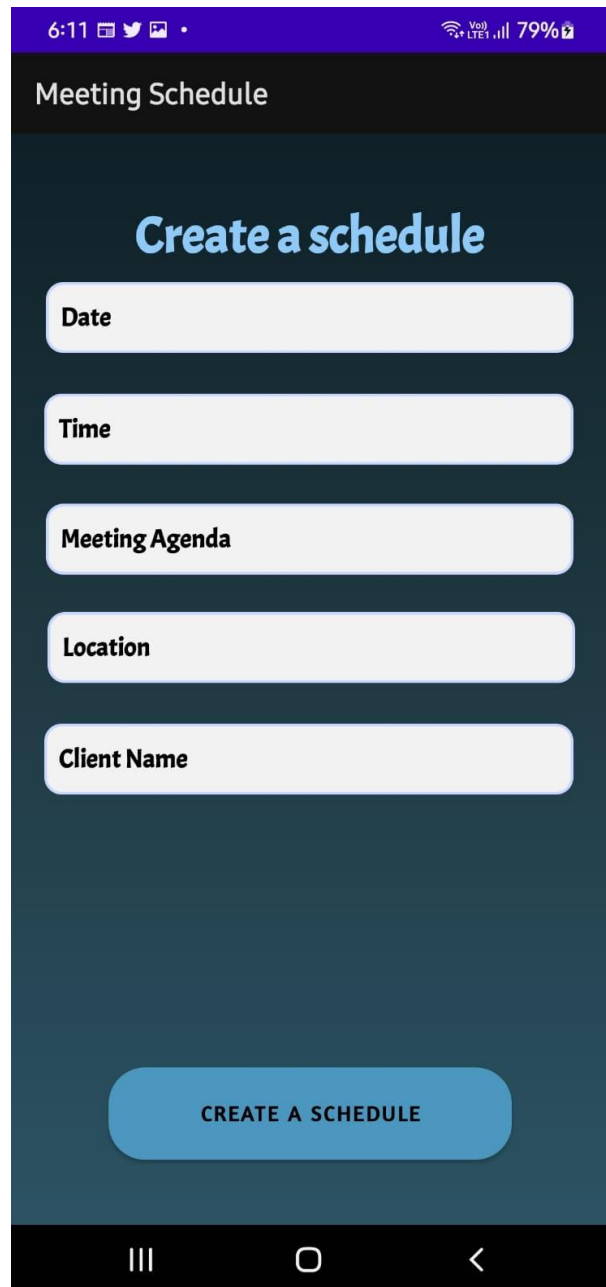
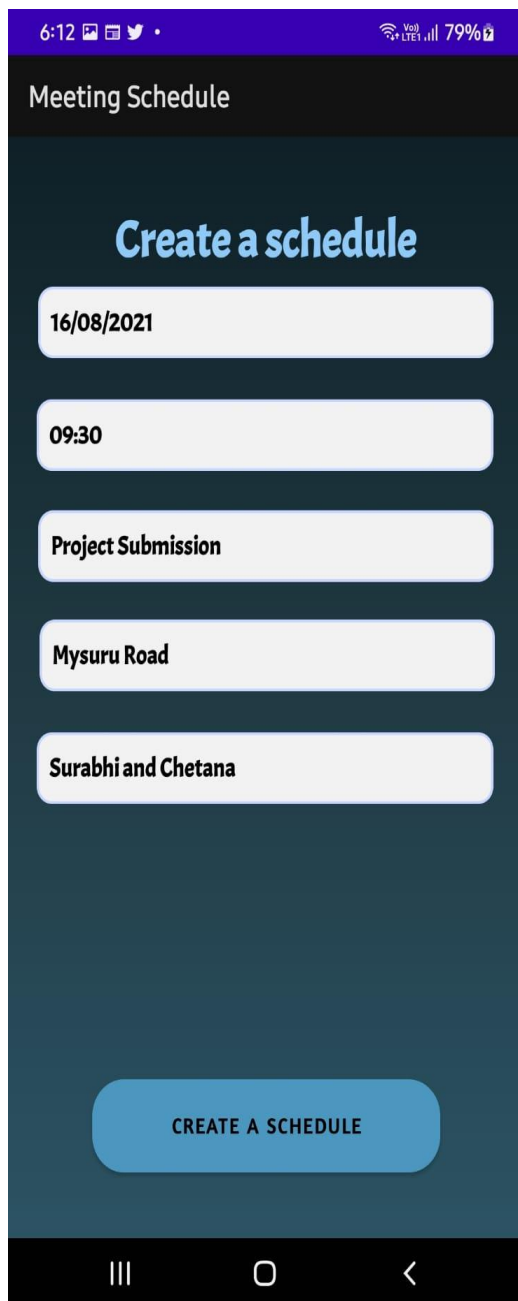
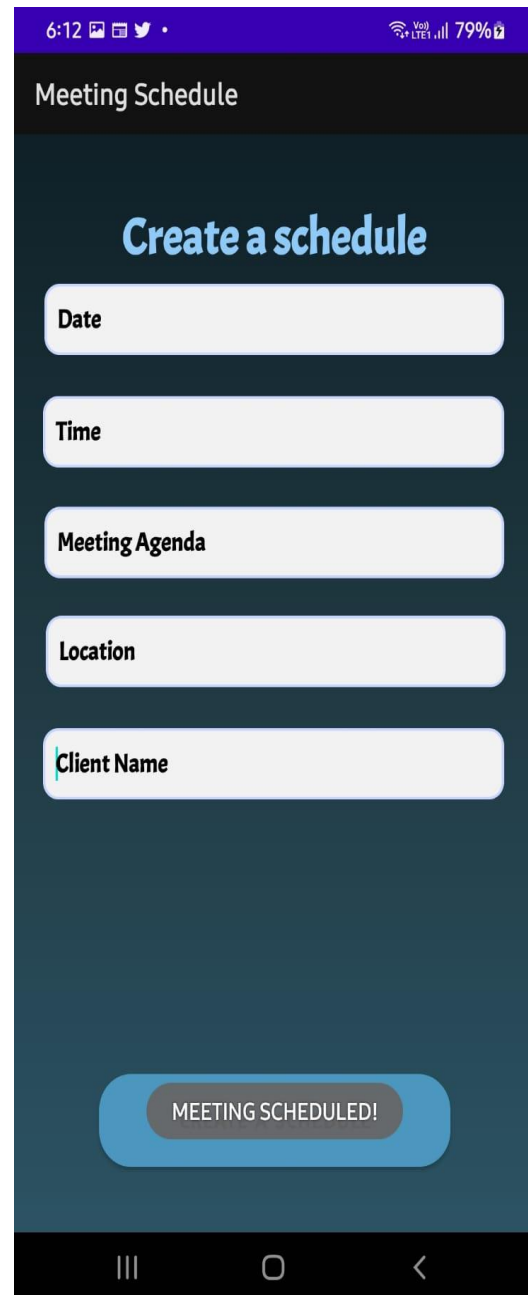


Fig 2: Schedule Page



A mobile application interface for scheduling a meeting. The screen has a dark blue background. At the top, a status bar shows the time 6:12, signal strength, and 79% battery. Below the status bar is a header with the text "Meeting Schedule". The main content area features the heading "Create a schedule" in light blue. Below this heading are five white input fields with rounded corners, each containing a label and a value: "16/08/2021", "09:30", "Project Submission", "Mysuru Road", and "Surabhi and Chetana". At the bottom of the form is a large blue button with the text "CREATE A SCHEDULE". The bottom of the screen shows a black navigation bar with three white icons: a hamburger menu, a circle, and a back arrow.

**Fig 3: Scheduling a Meeting**



A mobile application interface showing the result of scheduling a meeting. The screen has a dark blue background. At the top, a status bar shows the time 6:12, signal strength, and 79% battery. Below the status bar is a header with the text "Meeting Schedule". The main content area features the heading "Create a schedule" in light blue. Below this heading are five white input fields with rounded corners, each containing a label and a value: "Date", "Time", "Meeting Agenda", "Location", and "Client Name". At the bottom of the form is a large blue button with the text "MEETING SCHEDULED!". The bottom of the screen shows a black navigation bar with three white icons: a hamburger menu, a circle, and a back arrow.

**Fig 4: Meeting Scheduled**

## **CONCLUSION**

Through the development of meeting schedule on Android platform, we get a clear understanding of overall process of the system. The core part of the music player is mainly composed of main interface, playlists, menus, play Settings, and song search. Grasping the development of the six parts, the music player has had the preliminary scale. Based on the function of the six categories, add some other small features.

This design of music player based on Android system requires elaborate design of the meeting schedule framework, by adopting Eclipse3.5 + Java language as technical support of this system, with the Android plug-in tools, and combination of Android SDK3.6 version lead to the comprehensive and smoothly design and development of the mobile terminal.

## **FURTHER SCOPE**

Scope of further improvement:

1. We can make the app to have voice inputs
2. We can add the 24hrs and 12hrs format as optional
3. We can add the name of person who created the reminder/schedule

## **BIBLIOGRAPHY**

- i.** Google Developer Training, "Android Developer Fundamentals Course – Concept Reference", Google Developer Training Team, 2017.
- ii.** Erik Hellman, "Android Programming – Pushing the Limits", 1st Edition, WileIndia Pvt Ltd, 2014. ISBN-13: 978-8126547197
- iii.** Dawn Griffiths and David Griffiths, "Head First Android Development", 1st Edition, O'Reilly SPD Publishers, 2015. ISBN-13: 978-9352131341
- iv.** Bill Phillips, Chris Stewart and Kristin Marsicano, "Android Programming: The Big Nerd Ranch Guide", 3rd Edition, Big Nerd Ranch Guides, 2017. ISBN-13: 978-0134706054
- v.** Various open source materials from Internet.
- vi.** Training notes.
- vii.** Some requirements are gathered through various books from library.

## **WEBSITES:**

- i.** <https://developer.android.com/studio>
- ii.** <https://stackoverflow.com/questions/tagged/android>

## **DECLARATION**

We student of 6<sup>th</sup> semester BE, Computer Science and Engineering College hereby declare that project work entitled “Meeting Schedule” has been carried out by us at City Engineering College, Bangalore and submitted in partial fulfilment of the course requirement for the award of the degree of Bachelor of Engineering in Computer Science and Engineering of Visvesvaraya Technological University, Belgaum, during the academic year 2020-2021.

We also declare that, to the best of our knowledge and belief, the work reported here does not form the part of dissertation on the basis of which a degree or award was conferred on a earlier occasion on this by any other student.

Date:

Place: Bangalore

SURABHI GR  
(1CE18CS084)

CHETANA NATH  
(1CE18CS061)