# Scientific Research and Development of Mobile Application for Android Platform

Dimple Chawla*
Rahul Aggarwal**
Akansha Aggarwal***

**Abstract**

In recent years, the emergence of smart phones has changed the definition of mobile phones. Mobile phone is no longer just a communication tool, but also an essential part of the people's communication and daily life. Various applications added unlimited fun for people's lives. It is certain that the future of the network will be the mobile terminal. Now the android system in the electronics market is becoming more and more popular, especially in the Smartphone industry. Because of the open source, some of the development tools are free, so there are plenty of applications generated. This greatly inspired the people to use the android system. Android is a mobile operating system (OS) currently developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets. Android's user interface is mainly based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a virtual keyboard for text input.

The purposes for developing this android application for providing the services to the user. It maybe for providing information, communication, Image and video making, Entertainment, Recharge Bills and Storing data. Also giving chance to entrepreneurs and developers make their own application for income source and reach an even broader audience base.

This paper gives a complete knowledge of how to start working on eclipse and develop an application and get it run on emulator.

**Keywords:** Android SDK, ADT plug-in, AVD manager, Eclipse-IDE, java/xml, Android Development Architecture, Activities.



## I. Introduction

To develop apps for Android, here to use a set of tools that are included in Android Studio or Eclipse. In addition to using the tools from Android Studio, also access most of the SDK tools from the command line. Developing with Android Studio is the preferred method because it can directly invoke the tools that required for developing applications.

**App Workflow:** The four basic steps for developing any application (with or without android studio) includes:

**Dimple Chawla***
Department of IT, DIAS, Rohini, Delhi

**Rahul Aggarwal****
Student, DIAS, Rohini, Delhi

**Akansha Aggarwal*****
Student, DIAS, Rohini, Delhi

**Environment Setup**: During this phase you install and set up your development domain.

**Project Setup and Development**: In this phase you set up and develop your Android Studio project and application modules, which have all of the source code and resource files for your respective application.

**Building, Debugging and Testing**: During this phase you build your project into a debuggable.apk package(s) that you can install and run on the emulator or an Android-powered device.

**Publishing**: During this phase you configure and build your application for release and distribute your application to users.

Applications: These are the basics of Android applications:

Android applications are composed of one or more application integrants (activities, services, content providers, and broadcast receivers)

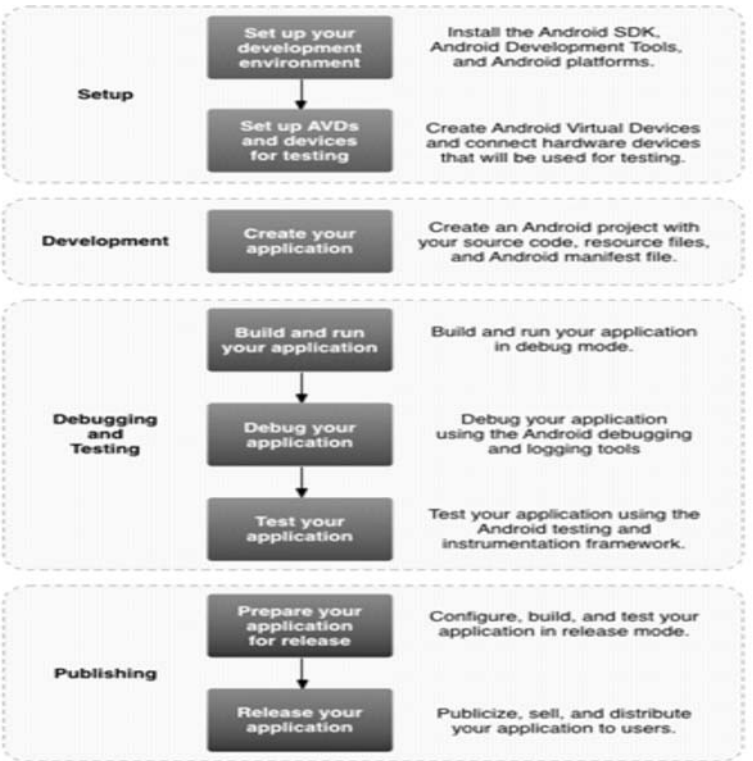


**Figure 1: The App Workflow**



Each component accomplishesa different role in the final application behavior, and each one can be triggered individually (even by other applications)

The manifest file must declare all components in the application and should also declare all application demands, such as the minimum version of Android necessary and any hardware configurations that are essential.

Non-code application resources (images, strings, layout files, etc.) should include options for different device configurations (such as different strings for different languages)

## Android System Architecture

**Applications:** Android gives a set of core applications pluging client, SMS program, calendar, maps, browser, contacts, and soon, all developed in Java.

**Application Framework:** The developer is allowed to access all the API framework of the core programs. The application framework simplifies the reuse of its segments. Any other app can release its functional components and all other apps can access and use this component by following the security aspects of the framework.

**Libraries and Android Run Time:** The library is divided in to two parts, Android Runtime and Android Library. Android Runtime is comprises of a Java Core Library and Dalvik virtual machine.

**Linux Kernel:** The kernel system service provided by androidinner nuclear layer is basedon Linux 2.6 kernel; operations like internal storage, process management, internet protocol, bottom-drive and other core service are all placed on Linux kernel.

**Eclipse:** Eclipse is an integrated development environment (IDE). It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in Java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages through the use of plugins.

## 1. Tools and Environment

Here we will discuss the installation details of software. Android SDK Tools, revision 20 or newer.

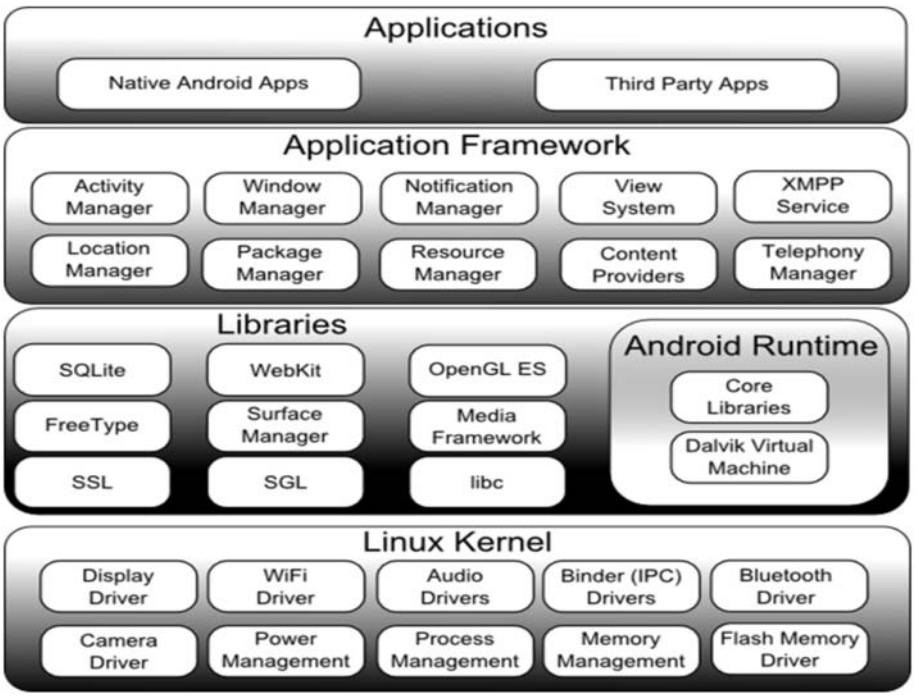


**Figure 2: Application Framework**



SDK Platform Android 3.0 (API 11).: The minimal platform supported by Java API is Android 2.2 (API 8). But for successful compilation the target platform should be set to Android 3.0 (API 11) or higher. It will not prevent them from running on Android 2.2.

Eclipse IDE: There is a list of Eclipse versions that are well matched with the Android SDK. In this paper we are using Eclipse 3.7 (Indigo).

ADT plug-in for Eclipse:Android Development Tools (ADT) is a plug-in for the Eclipse IDE that is designed to give us a strong, integrated environment in which to build Android applications.

ADT unroll the capabilities of Eclipse to let us rapidly set up new Android projects, create an application UI, add packages based on the Android Framework API, debug applications using the Android SDK tools, and even export signed (or unsigned) .apk files in order to spread the application. Developing in Eclipse with ADT is highly recommended and is the fastest way to get started. With the guided project setup it provides, as well as tools unification, custom XML editors, and debug output pane, ADT gives us an incredible elevation in developing Android applications.

Following paragraph will explain, how to download and install the ADT plug-in:

Start Eclipse and then select Help, now you have to click on → Install New Software.Now, Clickon Add, it is present on the top-right corner of your screen.

After doing all this, a Add Repository dialog box will appear, enter "ADT Plug-in" for the Name and the URL refer to Figure 3.

Now, Click OK and make sure that your system must be connected to the internet.In the Software dialog box, select the checkbox next to Developer Tools and then click next. A new window will appear and in that, we'll see a list of the tools to be downloaded. You have to click on Next. License agreement will appear, read and accept that and then click Finish. After this installation bar will appear, wait for its completion and once it is completed restart eclipse.

**AVD Manager:** The AVD Manager provides a graphical user interface with the help of which we can create and manage Android Virtual Devices (AVDs), which are required by the Android Emulator. For emulation, a device is defined. Select Window and then from the menu select Android AVD Manager.

**Table 1: Versions of Eclipse**

| Codename | Date | Platform version | Projects |
|---|---|---|---|
| N/A | 21 June 2004 | 3.0[14] | |
| N/A | 28 June 2005 | 3.1 | |
| Callisto | 30 June 2006 | 3.2 | Callisto projects |
| Europa | 29 June 2007 | 3.3 | Europa projects |
| Ganymede | 25 June 2008 | 3.4 | Ganymede projects |
| Galileo | 24 June 2009 | 3.5 | Galileo projects |
| Helios | 23 June 2010 | 3.6 | Helios projects |
| Indigo | 22 June 2011 | 3.7[Notes 1] | Indigo projects |
| Juno | 27 June 2012 | 4.2[15] | Juno projects |
| Kepler | 26 June 2013 | **4.3** | Kepler projects |
| Luna | 25 June 2014 (planned) | 4.4 | Luna projects |

Old version | Older version, still supported | Latest version | Future release



## Create a Project with Android Studio

You have to create a new project in the android studio. Android studio already gives you a opened project in the welcome screen but If you don't have a project opened, in the Welcome screen, clickon New Project but If you have a project opened the from the File menu, select the New Project. The *Create New Project* screen appears.

Now fill out the fields that appear on the screen, and then click on Next.

**Application Name** is the app name that appears to users. For this project, use "My First App."(or anything you like)

**Company domain** provides a qualifier that will be appended to the package name; Android Studio will remember this qualifier for each new project you create.

**Package name** is the fully qualified name for the project (following the same rules as those for naming packages in the Java programming language). Make sure that your package name must be unique across all packages installed on the Android system.

You can also **edit** this value independently from the application name or the company domain. **Project location** is the directory on your system that holds the project files.

Under the **Select form the factors on which your app will run on**, check on the box for **Phone and Tablet**(For **Minimum SDK**, select **API 8: Android 2.2 (Froyo)**.)Leave all of the other options (TV, Wear, and Glass) unchecked and then click on **Next.**

Activities An activity is one of the distinguishing features of the Android framework. Activities provide the user with access to your app, and there may be many activities. An application will usually have a main activity for when the user launches the application, another activity for when she selects some content to view, for example, and other activities for when she performs other tasks within the app.

After this **Add an activity to <***template***>**, select **Blank Activity** and then click on **Next**.

Under **Customize the Activity option**, change the **Activity Name** to *MyActivity*. The **Layout Name will** change to *activity_my*, and the **Title name will chang** to*MyActivity*. The **Menu Resource Name will be***menu_my*.now click on finish button and create your project.

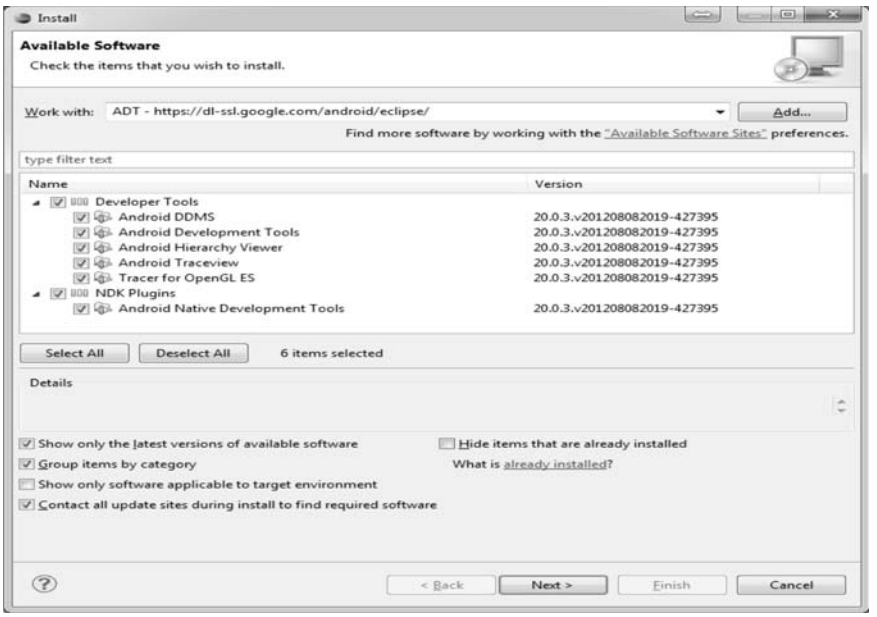


**Figure 3: ADT Plug-in wizard to install**



Developing an Audio Player

Step 1:Open the project's main layout file and replace its contents with the following layout:

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="vertical"
android:background="#FF330000"
tools:context=".MainActivity" >

<ListView
android:id="@+id/song_list"
android:layout_width="fill_parent"
android:layout_height="wrap_content" >
</ListView>

</LinearLayout>

**Step 2:** Store the title strings in the **res/values/strings.xml** file. The two items refer to drawable files. Create your own or use these two images to start with:

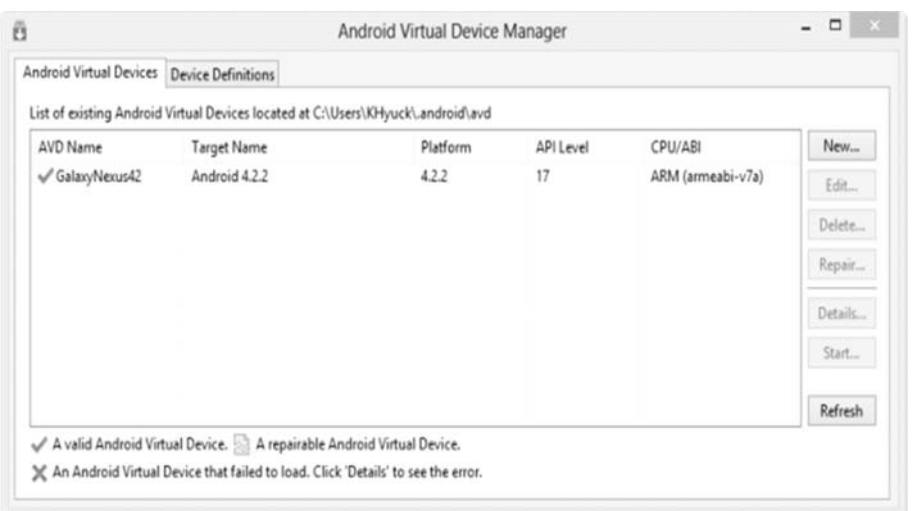


**Figure 4: Android Virtual Device Manager Wizard**

We will also use an icon to display in the playback notification. Create one now or use the one below:



The code will refer to the images using the names **rand**, **end**, and **play** so make sure that you use the same file names. Copy the images to your project's drawables folder(s). We will implement the actions later.

**STEP 3:**Open the main Activity class and add the following imports:

```
importjava.util.ArrayList;
importjava.util.Collections;
importjava.util.Comparator;
importandroid.net.Uri;
importandroid.content.ContentResolver;
importandroid.database.Cursor;
importandroid.widget.ListView;
```

Declare the following instance variables before the onCreate method:

```
private ArrayList<Song>songList;
private ListViewsongView;
```

We will store the songs in a list and display them in the ListView instance in the main layout. In onCreate, after setting the content view, retrieve the ListView instance using the ID we gave it in the main layout:

```
songView =
(ListView)findViewById(R.id.song_list);
```

Instantiate the list as shown below:

```
songList = new ArrayList<Song>();
```

Next, in the main Activity class declaration, after the existing methods, create a helper method to retrieve the audio file information:

Create a ContentResolver instance, retrieve the URI for external music files, and create a Cursor instance using the ContentResolver instance to query the music files:

```
ContentResolvermusicResolver =
getContentResolver();
Uri musicUri =
android.provider.MediaStore.Audio.Media.
EXTERNAL_CONTENT_URI;
Cursor musicCursor =
musicResolver.query(musicUri, null, null, null,
null);
```

Now we can iterate over the results, first checking that we have valid data:

```
if(musicCursor!=null
&&musicCursor.moveToFirst()){
  //get columns
inttitleColumn = musicCursor.getColumnIndex
(android.provider.MediaStore.Audio.Media.TITLE);
intidColumn = musicCursor.getColumnIndex
  (android.provider.MediaStore.Audio.Media._ID);
intartistColumn = musicCursor.getColumnIndex
  (android.provider.MediaStore.Audio.Media.ARTIST);
  //add songs to list
  do {
    long thisId =
musicCursor.getLong(idColumn);
    String thisTitle =
musicCursor.getString(titleColumn);
    String thisArtist =
musicCursor.getString(artistColumn);
songList.add(new Song(thisId, thisTitle,
thisArtist));
  }
  while (musicCursor.moveToNext());
}
```

We first retrieve the column indexes for the data items that we are interested in for each song, then we use these to create a new Song object and add it to the list, before continuing to loop through the results.

Back in onCreate, after the code we added, call this new method:

```
getSongList();
```

STEP 4: Back in the main Activity class, in the onCreate method after sorting the list, create a new instance of the Adapter class and set it on the ListView:

```
SongAdaptersongAdt = new SongAdapter(this,
songList);
songView.setAdapter(songAdt);
```

## II. Conclusion

Android is a full, open and free mobile device platform, with its powerful function and good user experience rapidly developed into the most popular mobile

operating system. This paper gives an overview of the different challenge and issues faced in android app development. It gives a detailed reference of a new music app.The experience of developing an android app is quite challenging, motivating as well as satisfying.

## References


1. Khawlah A. AI-Rayes, AiseZulalSevkli, Hebah F. AI-Moaiqel, Haifa M. AI-Ajlan, Khawlah M. AI-Salem, Norah I. AI-Fantoukh "A Mobile Tourist Guide for Trip Planning" IEEE MULTIDISCIPLINARY ENGINEERING EDUCATION MAGAZINE, VOL. 6, NO. 4, DECEMBER 2011
2. Priyanka Shah, RutaGadgil, Neha Tamhankar "Location Based Reminder Using GPS For Mobile (Android)" ARPN Journal of science and Technology ©2011-2012. VOL. 2, NO. 4, May 2012
3. Sumaiya Patel, Darshana Thakur, SujitSekhar. PriyanksDhamane "Lockme - Android Security Application" IJCER, VOL. 3, Issue. 3
4. http://developer.android.com