A Review and Basic Guidelines on Developing Android Applications

Saurabh Malgaonkar
Computer Engineering Department,
Mukesh Patel School of Technology Management & Engineering,
NMIMS University, Mumbai, India

Shailja Sumeet
Computer Engineering Department,
Mukesh Patel School of Technology Management & Engineering,
NMIMS University, Mumbai, India

Yash Radia
Computer Engineering Department,
Mukesh Patel School of Technology Management & Engineering,
NMIMS University, Mumbai, India

Nipun Philip
Computer Engineering Department,
Mukesh Patel School of Technology Management & Engineering,
NMIMS University, Mumbai, India

ABSTRACT

Users in today's world are on the move and they are using mobile application platformsto get there. Whether they use mobile phones, tablets, or other mobile devices they have all theinformation they need. That's why mobile apps are so much important in today's market. Android is now the most commonly used mobile operating system in the world. Android now has more users, more phones and more tablets worldwide than any other mobile operating system.

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

Keywords

Android SDK, Java/C++, Android Apps Development, Android Studio, IDE.

1. INTRODUCTION

Android operating system based mobile smartphones are very popular and are always in demand. It's got all the low-level "stuff" as well as the needed middleware to power and use an electronic device, and it is freely given away to anyone who wants to grab the code and build the operating system from it. [1]

When we say open source as mentioned above, it is free for everyone to use. Since we want to make an android application, we can make an application using the languages of C, C++ or Java. Java is more preferable.

Android also requires us to learn XML for the app design, understanding concepts of Android and using said concepts with the programming language.

Now to start making programs in Android we need an IDE (Integrated Development Environment).

Now the question is what is an IDE?

An IDE basically is an interface to help you write your programs. Take for example Turbo C, we use Turbo C to write a program and we are able to compile and run programs effectively on it. Thus Turbo C is an interface for user to let them compile and run the program.

In case of Android Development we use Android Studio or previously we used Eclipse.

Now we also come across a term called SDK (Software Development Kit). SDK only includes the necessary building blocks for developing applications. This includes frameworks, libraries, header files, whatever as well as compilers, debuggers, and various other tools, such as profilers, etc.[2]

Some SDK's have their own IDE so you don't need to download one separately. Basically a SDK is a necessity since it has all the tools required for an application. To make an android program you need an android SDK since to compile an android program you need a compiler, whose code or tools required for it resides in the SDK. In summary, an IDE is just an interface for using the compiler and so on, but a SDK is the kit which has the compiler. So without a SDK an IDE will not be there. Another term which we frequently come across is API. API stands for application program ***i***nterface. The API specifies how software components should interact and APIs are used when programming graphical user interface (GUI) components.[3] You can consider an API as the alternative "user interface" that software uses to interact with other software.
We humans are familiar with user interfaces that have fancy layouts with buttons, fonts, colors, graphics, etc; and most of that is unnecessary to a machine. Machines wouldn't open up a program or website, take a screenshot of that section of the monitor, and then try to parse its meaning visually like a humans do.
Machines just need a shorthand way to do things like checking the current weather or adding an event to your calendar. That's what an API provides.
APIs can be web-based, or specific to a platform. Google has APIs for search, calendars, translations, etc. Facebook and Twitter have APIs that allow software to automatically post status updates. Apple provides many APIs for building iPhone apps.

1.1 Java Language

Java is programming language. First generation of language was Machine language. Binary number 0 and 1, positive and negative. Than update machine language to second-generation language like Forton, Assembly etc. Now we are using 3rd Generation programming language. C, C++, Java etc. Java language has using everywhere for development application, operating system, Mathematical equation, algorithm, etc. Using java make an appropriate application. Develop an Android application in Java language, satisfactory and functionality of activities is good.

2. ARCHITECTURE

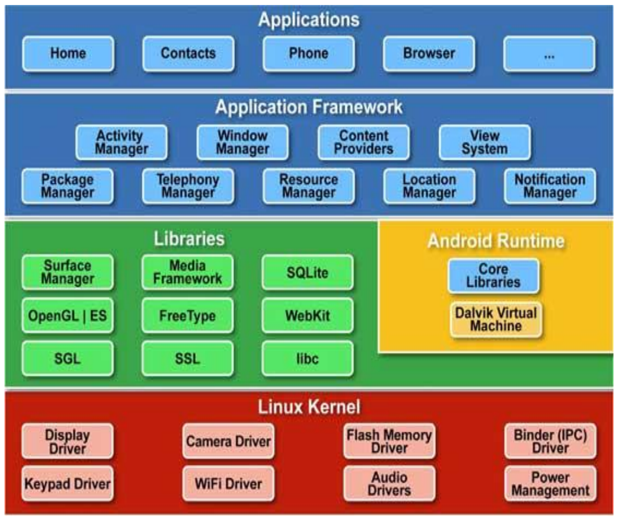


Figure 1: Android Architecture

2.1 Linux Kernel

The linux layer is responsible for the management of processes, memory as well as the input and output devices. The kernel efficiently controls and manages networking activities and connectivity among various devices through means of compatible interfacing.[4]

2.2 Libraries

The libraries provide support for the integration of many technologies like database systems, real time systems, networking frameworks etc. and their respective functionalities such as data transfer, security, streaming etc.[4]

A summary of the libraries are as follows:

- OpenGL (Open Graphics Library) It provides libararies for rendering 2D or 3D graphics.[4]
- SSL: For web security. It has no effect or use in our project.
- Surface Manager: Manages the display subsystem and handles the graphic layers for various applications.
- SGL: SGL stands for "Scalable Graphics Library" and is the graphics subsystem used by Android.
- WebKit: It is the browser engine used to display HTML content
- FreeType: It is the font engine.
- SqlLite: SqlLite is embedded into every Android device. Using an SqlLite database in Android does not require a setup procedure or administration of the database. You only have to define the SQL statements for creating and updating the database. Afterwards the database is automatically managed for you by the Android platform.
- Media Framework: Media framework provides different media codecs allowing the recording and playback of different media formats[4]

2.3 Android Runtime

It provides a virtual machine system known as 'Dalvik Virtual Machine' which provides the functionalities such as memory management and multithreading. It is an optimized version of the Java Virtual Machine.

2.4 Application Framework

It provides with many services which can be utilized by the developers to develop the Android applications.

A Summary of them is as follows:

2.4.1 Activity Manager

Users request to launch an application via a tap on an App icon from the home screen. The home screen as users know it is an application and the only one listening for onClick(). When this happens the launcher contact the ActivityManager. That is, request a handler through the **Binder** and call startActivity() method from the Activity Manager. [5]

2.4.2 Window Manager

Window manager is responsible for organizing the screen, applications don't get to decide that. The window manager allocates surfaces and decides where they go and how they are layered; it never touches their bits, which is up to the application.[6]

2.4.3 Content Providers

Content providers manage access to a structured set of data. They encapsulate the data, and provide mechanisms for defining data security.[7]

2.4.4 View System

Helps to design and develop interactive user interfaces.

2.4.5 Package Manager

Package Manager is an API that actually manages application install, uninstall, and upgrade. Telephony Manager: Provides information to the application about the telephony services available on the device such as status and subscriber information.

2.4.6 Resource Manager

Provides with all the essential resources required by the developer. It stores bitmaps strings etc.

2.4.7 Notification Manager

Provides a consistent and non-intrusive mechanism for signaling your users (example: when you are playing a game and someone sends you a text message, a notification does not terminate your game, instead, you might hear a sound).

2.5 Applications

Developers are given access to this layer for making the Android applications. Application development related contents are installed on this layer.

3. ANDROID APP COMPONENTS

Now that we have understood basic terminology and architecture of Android, lets understand some basic App components used in Android App components are the essential building blocks of an Android app.

An android application consists of four components. They are as follows:

3.1 Activities
An activity indicates the respective single screen instance through the user interface. Activity indicates the instance of the operation being performed by the user.

3.2 Services
Service component manages all the processes and the operations of the application with a extensive

support for multithreading which allows multiple activities to run despite of the main activity being currently performed by the user.[8] For example, user can keep music running in the background while performing other activities such as chat, e-mail etc.

3.3 Content Providers
It manages the overall data shared by various applications and provides proper interactivity among them. Through this component the applications have access to

the data as per the privileges granted by the content provider.

3.4 Broadcast Receivers
It is responds to the system calls, APIs or the calls generated by the applications. The interdependencies among the applications for performing the same operations is properly

handled by the Android operating system. Intent indicates the activation of three out of four components through asynchronous messaging. For example an intent can indicate a request for the current activity to capture a photo or to play a song. Broadcast receivers only broadcast the status of activity as required. E.g "Battery Low".

4. APP BUILD STEPS

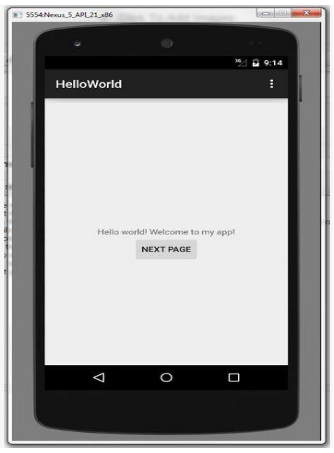
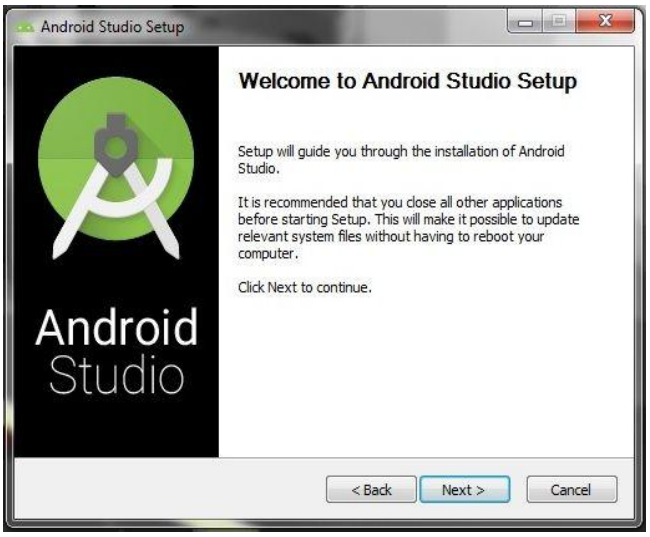


Figure 2: Sample output

The following steps are applicable once you have downloaded and installed Android Studio in your pc.

8 Steps:

Step 1: Android Studio Installation


Android Studio Setup

Welcome to Android Studio Setup

Setup will guide you through the installation of Android Studio.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

Figure 3: Android Studio Installation

1. Visit http://developer.android.com/sdk/index.html to download Android Studio.
2. Run the installer file to install Android Studio.
3. After successful installation follow the below instructions.

Step 2: Open a New Project

1. Run Android Studio.
2. Under the "Quick Start" menu, select "Start a new Android Studio project."
3. Click the "Create New Project" window that opens, name your project "HelloWorld".
4. If you choose to, set the company name as desired*. (Optional)
5. Check where the project file location is and change it if desired.
6. Click on "Next."
7. Make sure that "Phone and Tablet" is the only checked box.
8. For testing the app on your phone, make sure the minimum SDK is below your phone's operating system level.
9. Click on "Next."
10. Now Select "Blank Activity."
11. Click on "Next."
12. Leave all of the Activity name fields as they are.
13. Click "Finish."

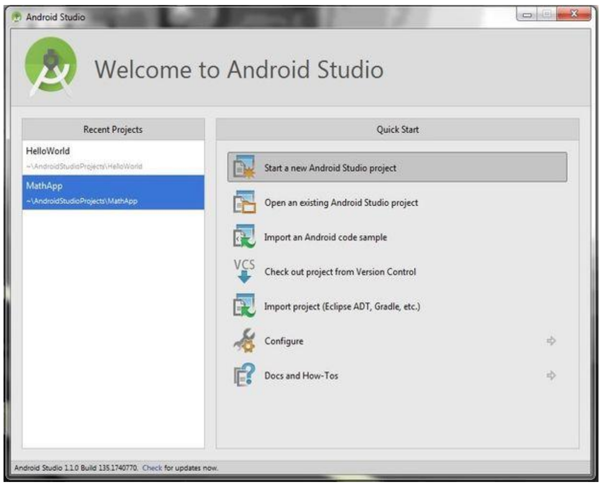

Android Studio
Welcome to Android Studio
Recent Projects
HelloWorld
~\AndroidStudioProjects\HelloWorld
MathApp
~\AndroidStudioProjects\MathApp
Quick Start
Start a new Android Studio project
Open an existing Android Studio project
Import an Android code sample
Check out project from Version Control
Import project (Eclipse ADT, Gradle, etc.)
Configure
Docs and How-Tos
Android Studio 1.1.0 Build 135.1740770. Check for updates now.

Figure 4: Creating a New Project (1/5)

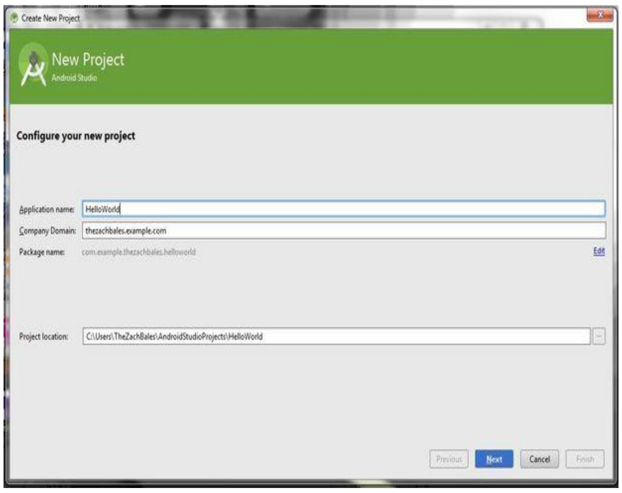

Create New Project
New Project
Android Studio
Configure your new project
Application name: HelloWorld
Company Domain: thezachbales.example.com
Package name: com.example.thezachbales.helloworld
Project location: C:\Users\TheZachBales\AndroidStudioProjects\HelloWorld

Figure 5: Creating a New Project (2/5)

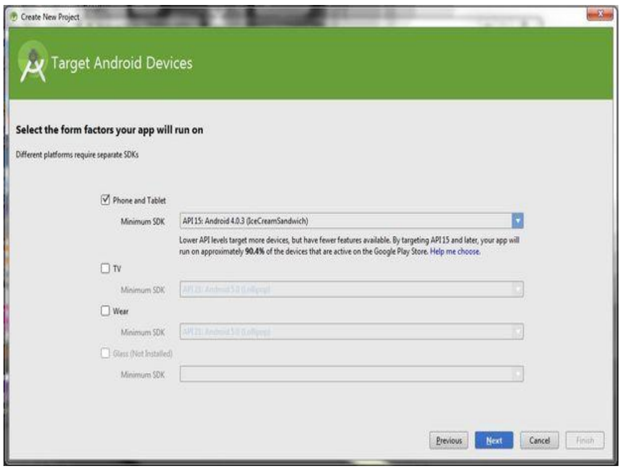

Create New Project
Target Android Devices
Select the form factors your app will run on
Different platforms require separate SDKs
[✓] Phone and Tablet
Minimum SDK: API 15: Android 4.0.3 (IceCreamSandwich)
Lower API levels target more devices, but have fewer features available. By targeting API 15 and later, your app will run on approximately **90.4%** of the devices that are active on the Google Play Store. Help me choose.
[ ] TV
Minimum SDK: API 21: Android 5.0 (Lollipop)
[ ] Wear
Minimum SDK: API 21: Android 5.0 (Lollipop)
[ ] Glass (Not Installed)
Minimum SDK

Figure 6: Creating a New Project (3/5)

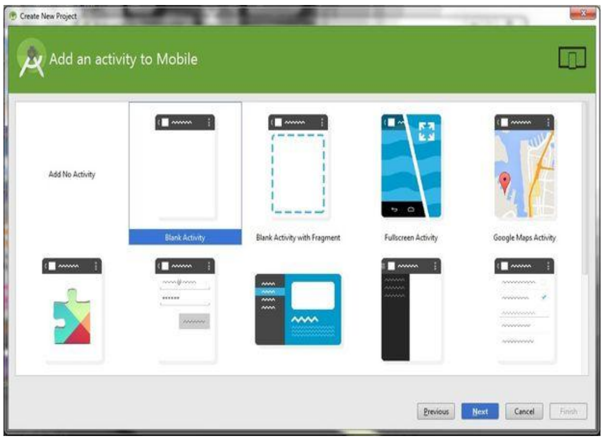


Figure 7: Creating a New Project (4/5)

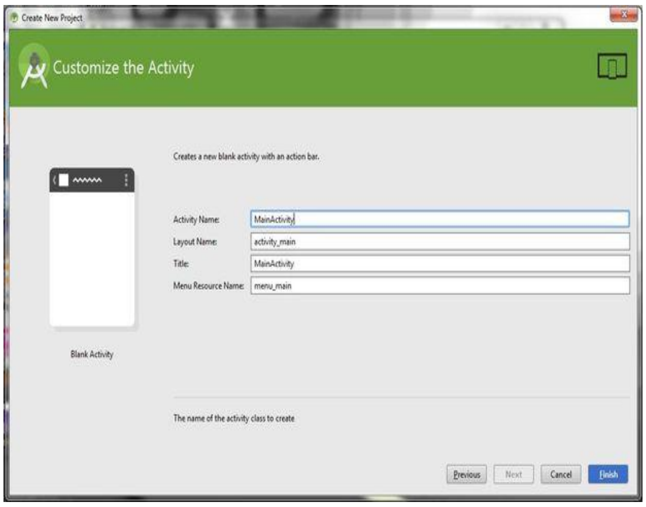

Create New Project
Customize the Activity
Creates a new blank activity with an action bar.
Blank Activity
Activity Name: MainActivity
Layout Name: activity_main
Title: MainActivity
Menu Resource Name: menu_main
The name of the activity class to create

Figure 9: Creating a New Project (5/5)

Step 3: Welcome Message for your app

1. Navigate to the activity_main.xml tab if it is not already open.
2. Make sure that the Design tab is open on the activity_main.xml display.
3. Click and drag the "Hello, world!" from the upper left corner of the phone display to the center of the screen.
4. In the project file system on the left side of the window, open the values folder.
5. In the values folder, double-click the strings.xml file.
6. In this file, find the line "Hello world!".
7. After the "Hello world!" message, add "Welcome to my app! (Or any message of your choice)"
8. Navigate back to the activity_main.xml tab.
9. Make sure that your centered text now reads "Hello world! Welcome to my app!"

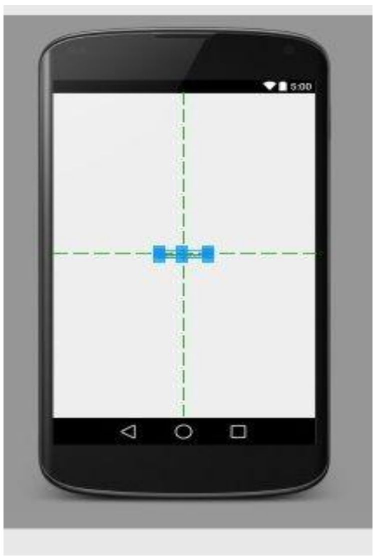
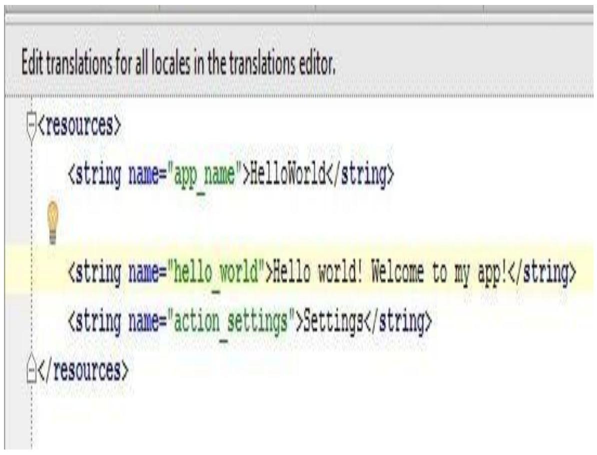


Figure 8: Preview Screen



Edit translations for all locales in the translations editor.

```
<resources>
    <string name="app_name">HelloWorld</string>

    <string name="hello_world">Hello world! Welcome to my app!</string>
    <string name="action_settings">Settings</string>
</resources>
```

Figure 10: Code to edit the welcome message

Step 4: Adding a Button in your app

1. Navigate to the Design tab of the activity_main.xml display.
2. In the Palette menu to the left of the phone display, find Button (under the heading Widgets).
3. Click and drag Button to be centered underneath your welcome message.
4. Make sure your button is still selected.
5. In the Properties menu (on the right side of the window), scroll down to find the field for "text."
6. Change the text from "New Button" to "Next Page."

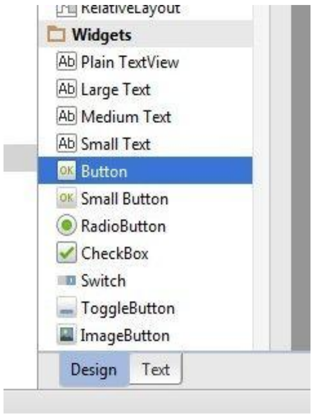
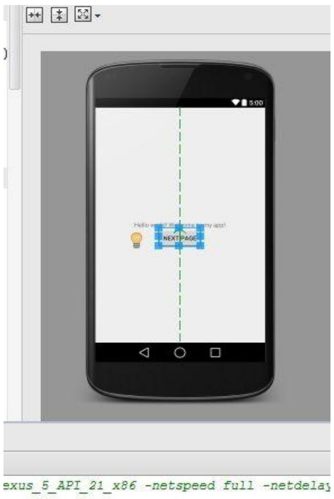


Figure 11: Additional Options



```
exus_5_API_21_x86 -netspeed full -netdelay
```

Figure 12: Preview Screen

Step 5: Creating an Additional Activity in your app

1. At the top of the project's file system tree, right click on "app."
2. Navigate through to New > Activity > Blank Activity.
3. Change the name of this activity to "SecondActivity".
4. Click "Finish."
5. Make sure you are in the Design view of activity_second.xml.
6. Drag the text box in the upper left of the phone display down to the center as you did on the Main Activity.
7. With the text box still selected, find the "id" field in the Properties menu on the right, and set it to "text2".
8. Open strings.xml again.
9. Add a new line under "Hello world! Welcome to my app!" that reads "Welcome to the second page!".
10. Navigate back to activity_second.xml.
11. Select the text box again.
12. In the Properties pane, set the "text" field to "@string/second_page".

13. Make sure that the text box now reads "Welcome to the second page!" and is in the center of the screen in the phone display.

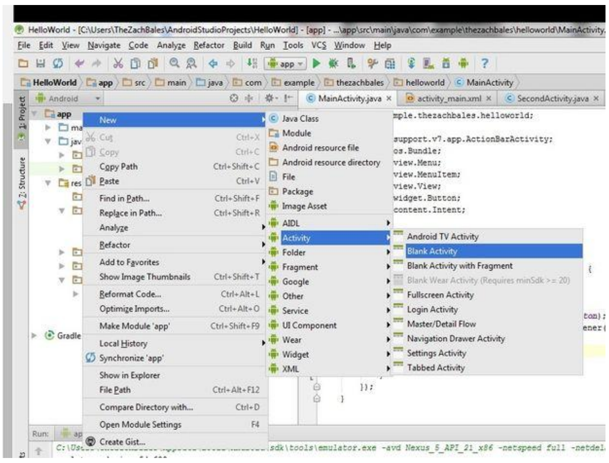
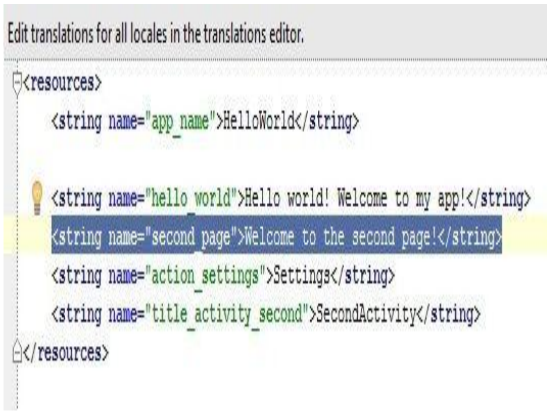

HelloWorld - [C:\Users\TheZachBales\AndroidStudioProjects\HelloWorld] - [app] - ...\app\src\main\java\com\example\thezachbales\helloworld\MainActivity.
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
HelloWorld app src main java com example thezachbales helloworld MainActivity
MainActivity.java activity_main.xml SecondActivity.java
New
Cut Ctrl+X
Copy Ctrl+C
Copy Path Ctrl+Shift+C
Paste Ctrl+V
Find in Path... Ctrl+Shift+F
Replace in Path... Ctrl+Shift+R
Analyze
Refactor
Add to Favorites
Show Image Thumbnails Ctrl+Shift+T
Reformat Code... Ctrl+Alt+L
Optimize Imports... Ctrl+Alt+O
Make Module 'app' Ctrl+Shift+F9
Local History
Synchronize 'app'
Show in Explorer
File Path Ctrl+Alt+F12
Compare Directory with... Ctrl+D
Open Module Settings F4
Create Gist...
Java Class
Module
Android resource file
Android resource directory
File
Package
Image Asset
AIDL
Activity
Folder
Fragment
Google
Other
Service
UI Component
Wear
Widget
XML
Android TV Activity
Blank Activity
Blank Activity with Fragment
Blank Wear Activity (Requires minSdk >= 20)
Fullscreen Activity
Login Activity
Master/Detail Flow
Navigation Drawer Activity
Settings Activity
Tabbed Activity
Gradle
Run:

Figure 13: Navigating Menu

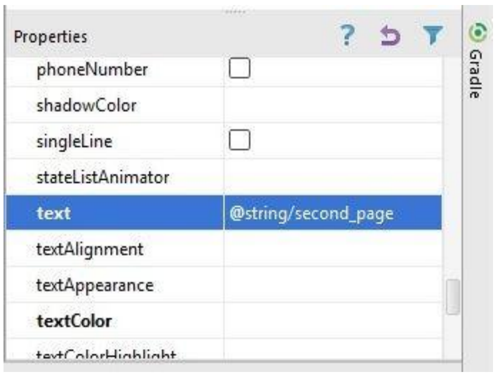


Edit translations for all locales in the translations editor.

```
<resources>
    <string name="app_name">HelloWorld</string>

    <string name="hello_world">Hello world! Welcome to my app!</string>
    <string name="second_page">Welcome to the second page!</string>
    <string name="action_settings">Settings</string>
    <string name="title_activity_second">SecondActivity</string>
</resources>
```

Figure 14: Code for addition of second page



Properties

phoneNumber	☐
shadowColor	
singleLine	☐
stateListAnimator	
text	@string/second_page
textAlignment	
textAppearance	
textColor	
textColorHighlight	

Figure 15: Properties

Step 6: Making it Interactive ("onClick")

1. Select the MainActivity.java tab along the top of the work environment.

2. Add the following lines of code at the end of the onCreate method:

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

button.setOnClickListener(new View.onClickListener()

{ @Override

public void onClick(View v)

{

goToSecondActivity();

}

});

3. Add the following method to the bottom of the *MainActivity class:*

private void goToSecondActivity()

{

Intent intent = new Intent(this, SecondActivity.class);

startActivity(intent);

}

4. Click the + next to import at the third line of MainActivity.java to expand the import statements.

5. Add the following to the end of the import statements if they are not already there:

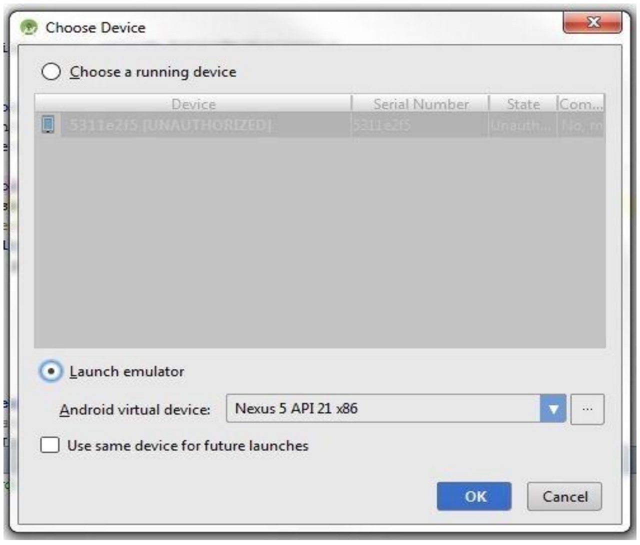
import android.content.Intent;

import android.view.View;

import android.widget.TextView;

Step 7: Testing the App

1. Click the green play symbol from the toolbar at the top of the Android Studio window.
2. When the "Choose Device" dialog appears (this may take a few moments), select the "Launch emulator" option.
3. Click OK.
4. When the emulator opens (this too could take a while), the app will automatically launch the app upon the virtual phone being unlocked.
5. Make sure that all of your text displays correctly and that the button takes you to the next page.



Choose Device

Choose a running device

Device	Serial Number	State	Com...
5311e2f5 [UNAUTHORIZED]	5311e2f5	Unauth...	No, m

[✓ Launch emulator]

Android virtual device: Nexus 5 API 21 x86

☐ Use same device for future launches

Figure 16: Testing through emulation

Step 8: And it's READY!
From here you have the cursory knowledge you need to go on to learn all there is to know about Android application development.

Case Study

Our system is divided into four layers, namely the UI layer, logic layer, App interface layer, the network access layer. Figure 10 is system architecture. UI layer is responsible for displaying the various forms of the system and the coordination between the various forms of invocation logic. The logical layer core control scheduling module is used to access the data transferred by UI, tasks need to be performed are calling App interfaces, accessing network data, returning message, refreshing UI, etc Network Access Layer is responsible for the system and the server's network connection and data transfer[9].

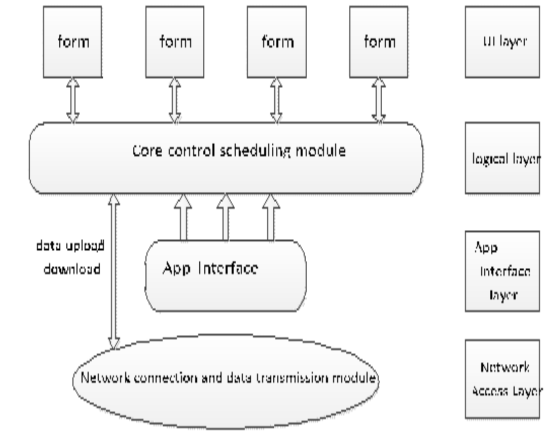


Figure 37: System Architecture

Authentication needs to be done before App SDK interacts with the server. SDK provides a class, when start the program,

the following code needs to be executed to create an App object, and set App Key, App Secret and URL.

App = App.getInstance(); // Create App object App.setupConsumerConfig(Consumer.consumerKey, Consumer.consumerSecret); // set App Key, App Secret App.setRedirectUrl(Consumer.redirectUrl); // Set the redirect URL

App.authorize(activity,newAuthDialogListenerImpl(activity)) ; After executing the above code, the login interface will appear, enter user name and password, after that, click the Login button. If it is the first time of login, the Authorization page appears. The main screen will show up after the user login.

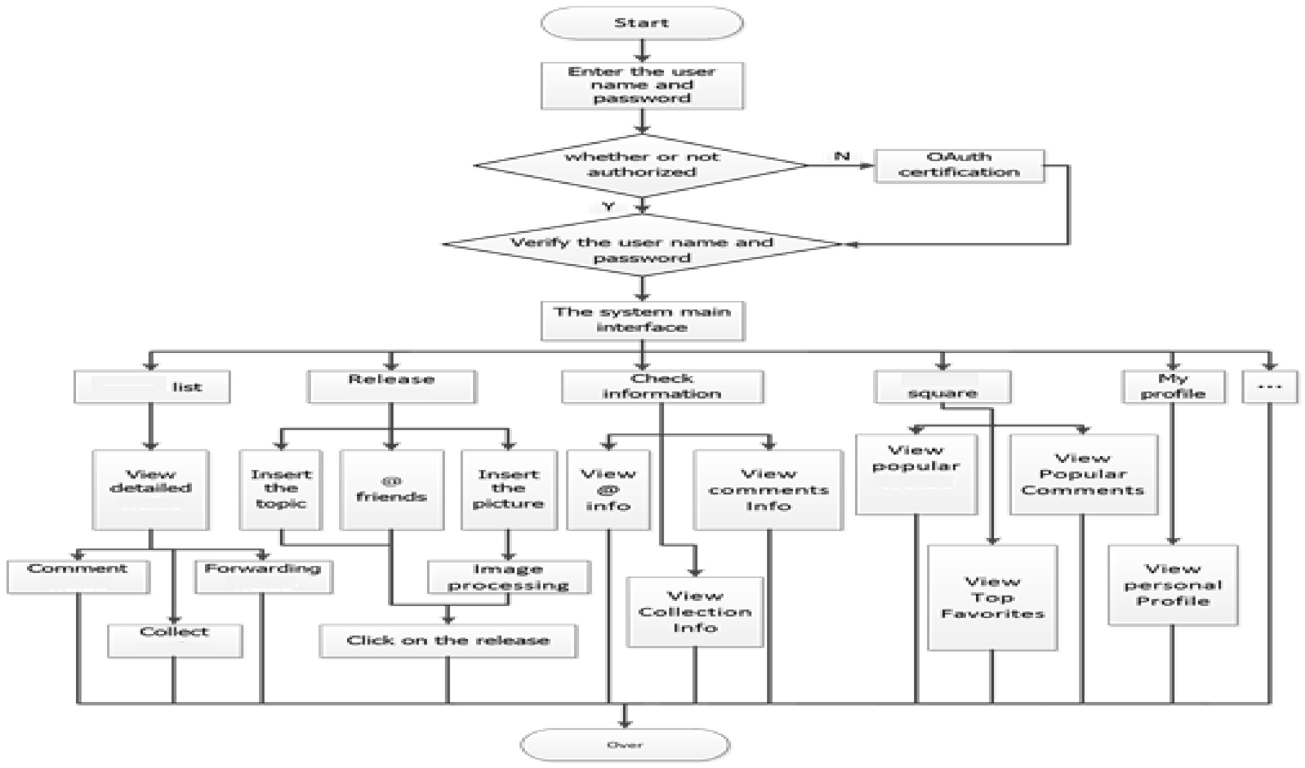


Figure 18: Flowchart

Specific functions of this system development are based on Android SDK, calling its wrapper classes to complete the corresponding task. For example, the main interface is divided into three parts: the top is a toolbar, the middle area is a ListView, bottom is the button bar. App List is in the middle of the main interface part, displayed through the ListView. As long as App data was obtained from the service side, it will be shown directly on the ListView control through the Adapter. The function of posting is achieved through AppManager.update method. This method can submit text and message containing pictures. The App browse window class is AppViewer and the interface layout file is app_viewer.xml.

5. REFERENCES

[1] ”Android Central”, http://www.androidcentral.com/what-android, January 2016.

[2] ”DifferencebetweenIDESDK”.http://bytes.com/topic/software-development/answers/910050-what-difference-between-ide-sdk., January 2016.

[3] ”AndroidAPI”,http://www.webopedia.com/TERM/A/API.html, January 2016.

[4] “AndroidArchitecture”,http://www.eazytutz.com/android/android-architecture/, January 2016.

[5] ”AndroidSupportFiles”,http://anatomyofandroid.com/2013/10/16/activity-manager/, February 2016.

[6] ”WindowsAndroidManager”,http://stackoverflow.com/questions/14952574/windows-manager-in-android-architecture, February 2016.

[7] “AndroidPackage”, http://java.dzone.com/articles/depth-android-package-manager, February 2016.

[8] ”AndroidDeveloperGuide”,http://developer.android.com/guide/components/fundamentals.html, March 2016.

[9] Research and Development of Mobile Application for AndroidPlatform“”,http://www.sersc.org/journals/IJMUE/vol9_no4_2014/20.pdf, March 2016, International Journal of Multimedia and Ubiquitous Engineering, Vol.9, No.4 (2014), pp.187-198.

6. AUTHOR PROFILE

Saurabh Malgaonkar is an assistant professor in the computer engineering department of Mukesh Patel School of Technology Management & Engineering, NMIMS University, Mumbai India.

Areas of Interests: Computer Networks, Data Mining Email: **Shailja Sumeet** is an assistant professor in the computer engineering department of Mukesh Patel School of Technology Management & Engineering, NMIMS University, Mumbai India.

Areas of Interests: Database Systems, Artificial Intelligence, Parallel Processing, Programming, Email: **Nipun Philip** is a student from the computer engineering department of Mukesh Patel School of Technology Management & Engineering, NMIMS University, Mumbai India. Areas of Interests: Software Development, Programming,

Yash Radia is a student from the computer engineering department of Mukesh Patel School of Technology Management & Engineering, NMIMS University, Mumbai India. Areas of Interests: Web Development, Human ComputerInterfacing