GALGOTIAS UNIVERSITY
(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Goggle-virtual Tryon

A Report for the Evaluation 3 of Project 2

Submitted by

Utkarsh Pandey

(1613101796 / 16SCSE101151)

in partial fulfillment for the award of the degree Bachelor of Technology

IN

Computer Science and Engineering

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

Under the Supervision of
Dr. Pallavi Murghai Goel(Associate Professor)

APRIL / MAY- 2020

SCHOOL OF COMPUTING AND SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this project report "Goggle-Tryon**"** is the bonafide work of **"Utkarsh Pandey(1613101796)"** who carried out the project work under my supervision.

SIGNATURE OF HEAD

Dr. Raju Shanmugam

Professor & Dean,

School of Computing Science & Engineering

.**SIGNATURE OF SUPERVISOR**

Dr.Pallavi Murghai Goel

Associate Professor

School of Computing Science &Engineering

1.Abstract

In Android is the mobile operating system designed mainly for touchscreen Devices. It is developed by the Open Handset Alliance which came as an upgrade from Linux systems. It is powered by Google and the project uses a Google Android Mobile Software Development Kit SDK 1.0 for testing an application. Any mobile user using this project is aimed at developing an application that incorporates both location and data call technology. To enable the Simulation of a State, which approximates and 30 Foreign Application Priority Data actual eyeglasses wearing State, by integrating previously inputted prescription data, lens material data and lens design, with chosen frame shape, enabling the easiest and most accurate determination of eyeglasses that are going to be ones' preference. it will provide more information related to frame and that will make choice easier by using a multi-media catalog library comprising photographs of frames with various color and design options and frame-related text. Further, facilitates opportunity by actively suggesting a pair comparison and on the selection Screen, which will compare Side-by Side of the same eyeglasses on the face, Try-on Screens will display Selected eyeglasses.

Techniques and functionalities which provide interactions between a representation Of a Selected pair of glasses and a fully textured 3D face model are disclosed. According to one personification, an interactive platform is displayed to allow a user to Select a pair of glasses from the digital catalog and try the Selected glasses on a user-provided image 3D face model. The interplay provided in the platform include contiguous adjustments of the glasses around the face model, various perspective views of the 3D. face with the glasses on, so that the user gets comfortable with the appearance on the face and other maquillage alternations to the Selected glasses. According to application, when the user finishes the “try-on” process, the information about the glasses can be sent to a manufacturer that can Subsequently produce pair of customized glasses for the client.

CHAPTER NO	TABLE OF CONTENTS	PAGE NO

CHAPTER 1

INTRODUCTION

1.1 BRIEF INTRODUCTION

1.1.1. What is Goggle?

Goggle is an application which is created to help people find an appropriate googgle suiting his/her personality from a wide range of goggles.

1.1.2. How do Goggle works?

Goggle is an android app which work for providing perefect goggle for an individual. it consist of huge collection of goggles from which a person can choose according to his/her face. once an application is installed on your phone you can test goggles on your face by either taking your picture from camera or by selecting a picture from your phone gallery.

1.1.3. How will Goggle tools save me time?

Once your data is loaded, our Goggle tools take no more than a few seconds to access. Our system will automatically provide you the range of goggles from which you can easily select according to your preference.

CHAPTER 2

SCOPE & OBJECTIVE OF PROJECT

2.1 PURPOSE

2.1.1. OBJECTIVE:

This project uses the Google Android platform to build an application where users can easily select goggles from variety of goggles. The objective of this app is to save time by choosing the goggles by trying it on their face sitting at home, User can also try it on any model and can also select picyure from gallery of phone and also select it from facebook. The most important information used in developing our Model Profiles is your feedback on what you expect from your new user. This is where generic benchmarking may fall short.

2.2. SCOPE:

Goggle project uses the Google Android platform to build an application where users can easily select goggles from variety of goggles. The objective of this app is to save time by choosing the goggles by trying it on their face sitting at home, User can also try it on any model and can also select picyure from gallery of phone and also select it from facebook. Our system compares the applicant's profile to the Model Profile and calculates the percentage match between the two. The two main domains are the Web application and the Mobile application. The users shall use the Android mobile in order to learn the problems, which were faced by the other users. This application benefits both users so that the problems can be addressed quickly and accounts for a friendly relationship between both the groups.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1.FUNCTIONALITY AND REQUIREMENTS

This section explains about the individual functionality and requirements of two major sections of this project

- Mobile Application
- Web Application

These requirements are identified from the sequence diagrams that are drawn to identify the flow of these two applications and the communication between them.

3.1.1 MOBILE APPLICATION:

Application requires service providers User Name and password. When a service provider starts this application a menu showing various options. If he/she selects the **fun app** list of all sessions will displayed , he/she can retrieve information of applications from these sessions and they can call, message or send mails to the userss . If he/she clicks on the **user details** then information of user will be displayed. When clicks on **the user name** then a window consists of various information or further fields where in you type name, last name, address and contact information will be displayed. When he/she clicks on **Email** then the user query the database and gets the information of the user in the area provided. Then a mail will send to the specified mail address.

3.1.2 WEB APPLICATION:

This application starts with a Flash intro and it later redirects the user into the web page where he can only login and use the services according to his account type. It concentrates on the functionality and requirements for different type of users. This application has 2 types of services

1. Models
2. Camera
3. Album

4. Facebook

These requirements are individually identified for each type of user and it is explained in detail as followed.

3.2. ASSUMPTIONS AND DEPENDENCIES:

1. The Mobile Application is dependent on the Web Application because the mobile application cannot work without the data from the web application and vice versa.
2. The default settings on the mobile application will reflect on the Google maps.
3. Settings such as frequently used zip code and the radius can be made default, thereby providing an easy way for the service provider.

4.Each Service provider having a Google Android phone.

Every service provider using this application shall possess a smart phone with Android Operating system on top of it.

3.2.1 CONSTRAINTS:

1. Huge data shall not be stored on the mobile phone. As a matter of fact, the mobile phone consists of limited memory and storing excess information is not at all a good design.

2. User of the mobile should have internet access. The Service provider should have the internet access in order to retrieve the service seeker's information. Whenever he/she updates his/her profile over the mobile, the changes should be synchronized with the web application.

3.3LESSONS LEARNED

3.3.1. IMPORTANCE OF RESEARCH:

We have used few technologies in our project, among them Android operating system is very important as our final application should run on this platform. We have spent many days learning Android as it was a new technology. Our application will be successfully built on this as we were able to use many built-in features of Android.

3.3.2. TEAM WORK:

By the end of this project we will end up as an effective and coordinating team as we understood the importance of the team work by the guidance of our mentor. Our team is a good combination of challenging and hardworking people. Throughout this project we have learnt a lot about team coordination, planning, presentation and developing personal attitude towards teamwork.

3.3.3. TIME MANAGEMENT:

To become successful, one must have good time management as it is considered as one of the important quality in the current competitive world. Keeping our mentor suggestions in mind we were able to implement and manage things in time. Meeting the various deadlines set by the instructor was tough and gave us a valuable experience of how to effectively manage time and as well the mentor’s expectations were sometimes very challenging and finally our project timeline was nearly accurate and we were following that from the initial stages onwards.

CHAPTER 4

DATABASE DESIGN

4.1. WHAT IS SQLITE

SQLite is an Open Source Database which is embedded into Android. SQLite supports standard relational database features like SQL syntax, transactions and prepared statements. In addition it requires only little memory at runtime (approx. 250 KByte).

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

4.1.2 WHAT IS SDK

A software development kit (**SDK**) is a collection of software development tools in one installable package. They ease creation of applications by having compiler, debugger and perhaps a software framework. They are normally specific to a hardware platform and operating system combination

4.2. SQLITE IN ANDROID

- SQLite is available on every Android device. Using an SQLite database in Android does not require any database setup or administration.
- 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.

- Access to an SQLite database involves accessing the filesystem. This can be slow. Therefore it is recommended to perform database operations asynchronously, for example via the AsyncTask class. .
- If your application creates a database, this database is saved in the directoryDATA/data/APP_NAME/databases/FILENAME.
- The parts of the above directory are constructed based on the following rules. DATA is the path which theEnvironment.getDataDirectory() method returns. APP_NAME is your application name. FILENAME is the name you specify in your application code for the database.

4.3. SQLITE ARCHITECTURE

4.3.1. Packages

The package android.database contains all general classes for working with databases.android.database.sqlite contains the SQLite specific classes.

4.3.2. SQLiteOpenHelper

- To create and upgrade a database in your Android application you usually subclass SQLiteOpenHelper. In the constructor of your subclass you call the super() method of SQLiteOpenHelper, specifying the database name and the current database version.
- In this class you need to override the onCreate() and onUpgrade() methods.
- onCreate() is called by the framework, if the database does not exists.
- onUpgrade() is called, if the database version is increased in your application code. This method allows you to update the database schema.
- Both methods receive an SQLiteDatabase object as parameter which represents the database.
- The database tables should use the identifier _id for the primary key of the table. Several Android functions rely on this standard.

4.3.3. SQLiteDatabase

- SQLiteDatabase is the base class for working with a SQLite database in Android and provides methods to open, query, update and close the database.
- More specifically SQLiteDatabase provides the insert(), update() and delete() methods.
- In addition it provides the execSQL() method, which allows to execute SQL directly.

- The object ContentValues allows to define key/values. The "key" represents the table column identifier and the "value" represents the content for the table record in this column. ContentValues can be used for inserts and updates of database entries.
- Queries can be created via the rawQuery() and query() methods or via the SQLiteQueryBuilder class .
- rawQuery() directly accepts an SQL statement as input.
- query() provides a structured interface for specifying the SQL query.
- SQLiteQueryBuilder is a convenience class that helps to build SQL queries.

CHAPTER 5

TECHNOLOGY USED

5.1. TECHNICAL DETAILS

5.1.1. TECHNOLOGIES USED:

This section focuses on technologies used in Web interface and Mobile Interface.

5.1.1.1. JAVA:

As the android platform understands Java the application was built on it.

5.1.1.2. XML:

XML is a simple, very flexible text format which was designed to carry data, not to display data.

5.1.1.3. Android:

The Android platform is a software stack for mobile devices including an operating system, middleware and key applications. Developers have full access to the same framework APIs used by the core applications. The application architecture is designed to simplify the reuse of components; any application can publish its capabilities and any other application may then make use of those capabilities. Developers can create applications for the platform using the Android SDK. Applications are written using the Java programming language and run on Dalvik, a custom virtual machine designed for embedded use, which runs on top of a Linux kernel.

In other words, Android is an operating system based on Linux with a Java programming interface.

- The Android Software Development Kit (Android SDK) provides all necessary tools to develop Android applications. This includes a compiler, debugger and a device emulator, as well as its own virtual machine to run Android programs.
- Android is currently primary developed by Google.
- Android allows background processing, provides a rich user interface library, supports 2-D and 3-D graphics using the OpenGL libraries, access to the file system and provides an embedded SQLite database.
- Android applications consists out of different components and can re-use components of other applications if these applications declare their components as available. This leads to the concept of a task in Android, an application can re-use other Android components to archive a task.

- For example you can write an Application which integrates the a map component and a camera component to archive a certain task.

ANDROID COMPONENTS

The followings gives a short overview of important Android components.

Activity

Activity represents the presentation layer of an Android application. A simplified (and slightly incorrect) description is that an Activity is a a screen. This is slightly incorrect as Activities can be displayed as dialogs or transparent. An Android application can have several Activities.

Views and ViewGroups

Views are user interface widgets, e.g. buttons or text fields. The base class for all Views isandroid.view.View. Views often have attributes which can be used to change their appearance and behavior.

A ViewGroup is responsible for arranging other Views e.g. they are layout manager. The base class for a layout manager is android.view.ViewGroups. ViewGroup also extends View. ViewGroups can be nestled to create complex layouts. You should not nestle ViewGroups too deeply as this has a negative impact on performance.

Intents

- Intents are asynchronous messages which allow the application to request functionality from other components of the Android systen, e.g. from Services or Activities. An application can call a component directly (explicit intent) or ask the Android system to evaluate registered components for a certain Intents (implicit intents). For example the application could implement

sharing of data via an Intent and all components which allow sharing of data would be available for the user to select. Applications register themselves to an intent via an IntentFilter.Intents allow to combine loosely coupled components to perform certain tasks.

Services

Services perform background tasks without providing an UI. They can notify the user via the notification framework in Android.

ContentProvider

ContentProvider provides an structured interface to data. Via a ContentProvider your application can share data with other applications. Android contains an SQLite database which is frequently used in conjunction with a ContentProvider to persists the data of the ContentProvider.

BroadcastReceiver

BroadcastReceiver can be registered to receives system messages and Intents. BroadcastReceiverwill get notified by the Android system if the specified situation happens. For example a BroadcastReceivercould get called once the system completed its boot process or if a phone call is received.

Every Android application runs in its own process

(*with its own instance of the Dalvik virtual machine*).

Whenever there's a request that should be handled by a particular component,

• Android makes sure that the application process of the component isrunning,

• starting it if necessary, and

• That an appropriate instance of the component is available, creating theinstance if necessary.

Application’s Life Cycle

A Linux process encapsulating an Android application is created for the application when some of its code needs to be run, and will remain running until

1. it is no longer needed, **OR**
2. the system needs to reclaim its memory for use by other applications.

An unusual and fundamental feature of Android is that an application process's lifetime is not directly controlled by the application itself.

Instead, it is determined by the system through a combination of

1. the parts of the application that the system knows are running,

2. how important these things are to the user, and

3. how much overall memory is available in the system.

Component Lifecycles

Application components have a **lifecycle**

1. A **beginning** when Android instantiates them to respond tointents through to an **end** when the instances are destroyed.

2. In **between**, they may sometimes be *active* or *inactive*, or -in thecase of activities-*visible* to the user or *invisible*.

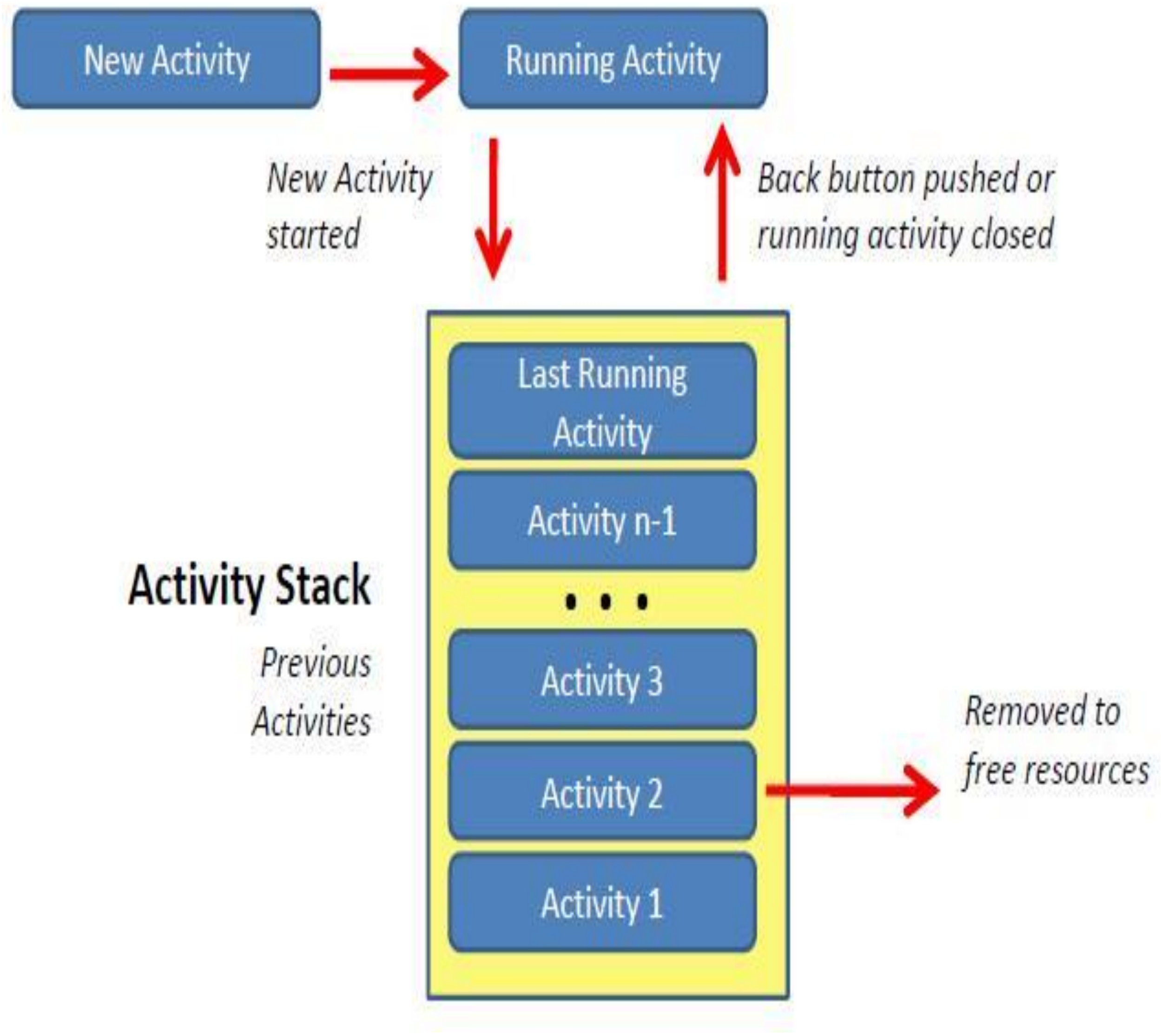
Start
Life as an Android Application:
Active / Inactive
Visible / Invisible
End

Activty Stack

• Activities in the system y are managed as an ***activity stack***.

• When a new activity is *started*, it is placed on the *top* of the stack and becomes the running activity -- the previousactivity always remains below it in the stack, and will notcome to the foreground again until the new activity exits.

• If the user presses the *Back Button* the next activity on thestack moves up and becomes active.



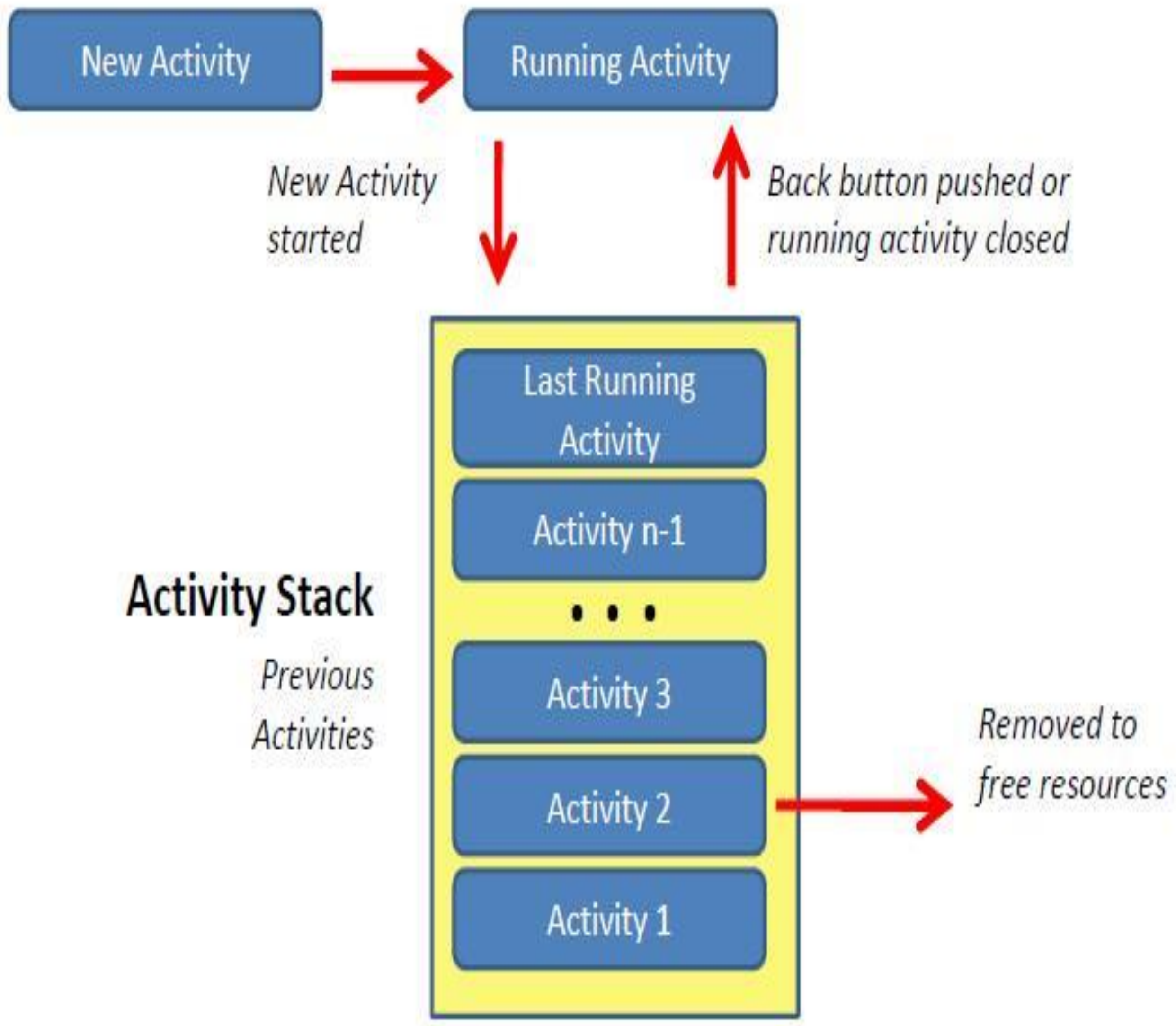
Life Cycle States

An activity has essentially three states:

1. It is ***active* or *running***

2. It is ***paused*** or

3. It is ***stopped*** .



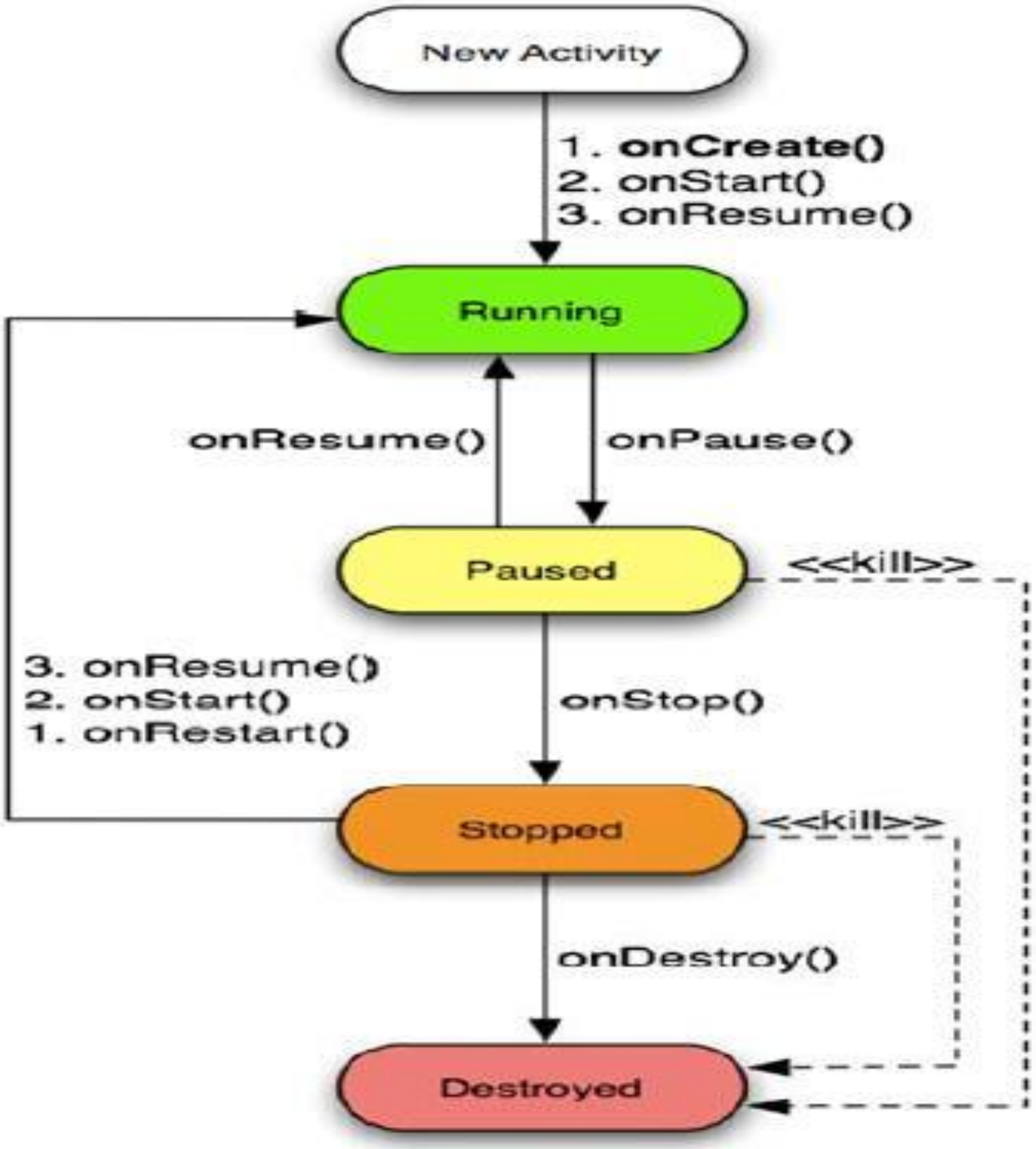
Life Cycle States

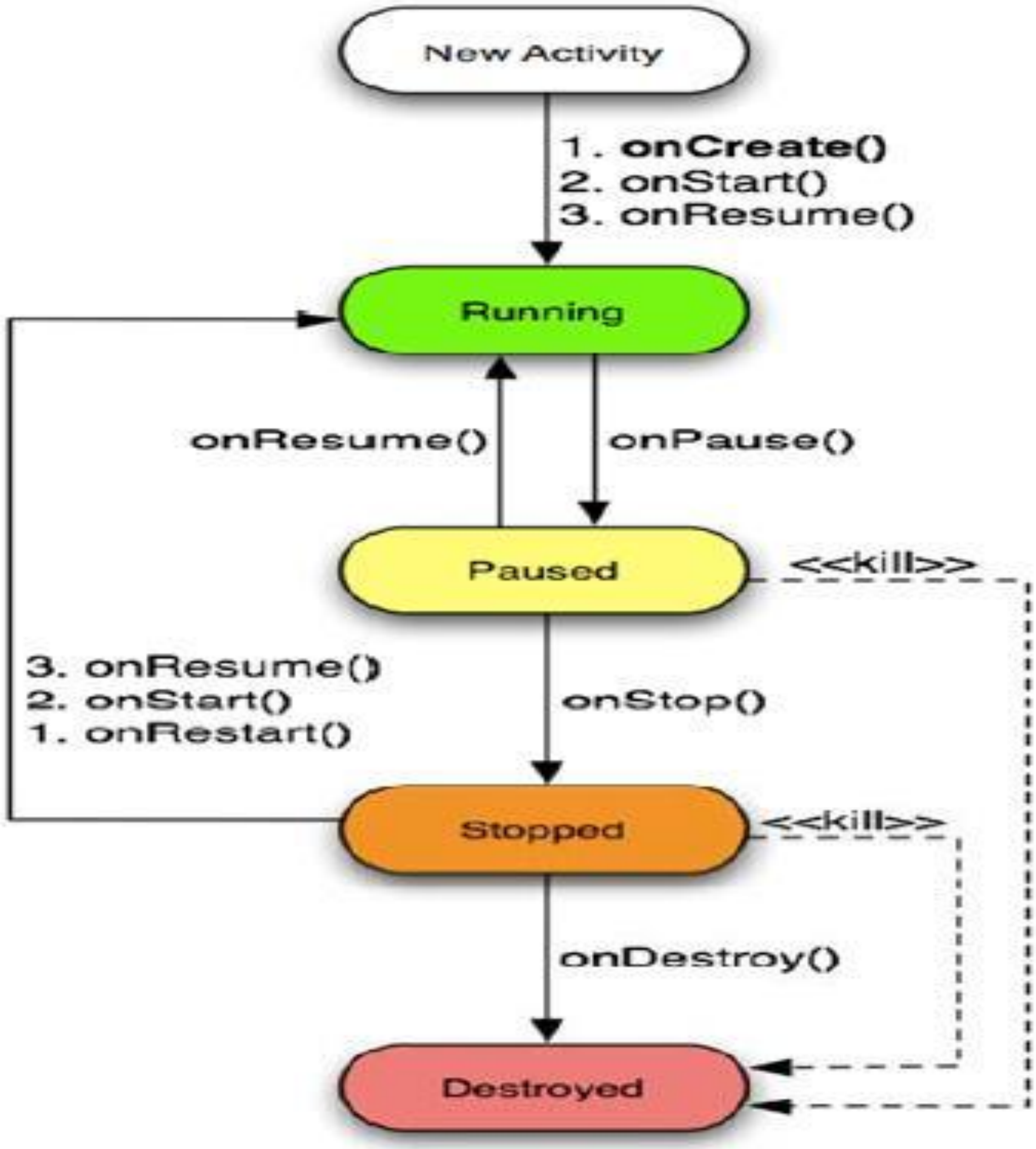
An activity has essentially three states:

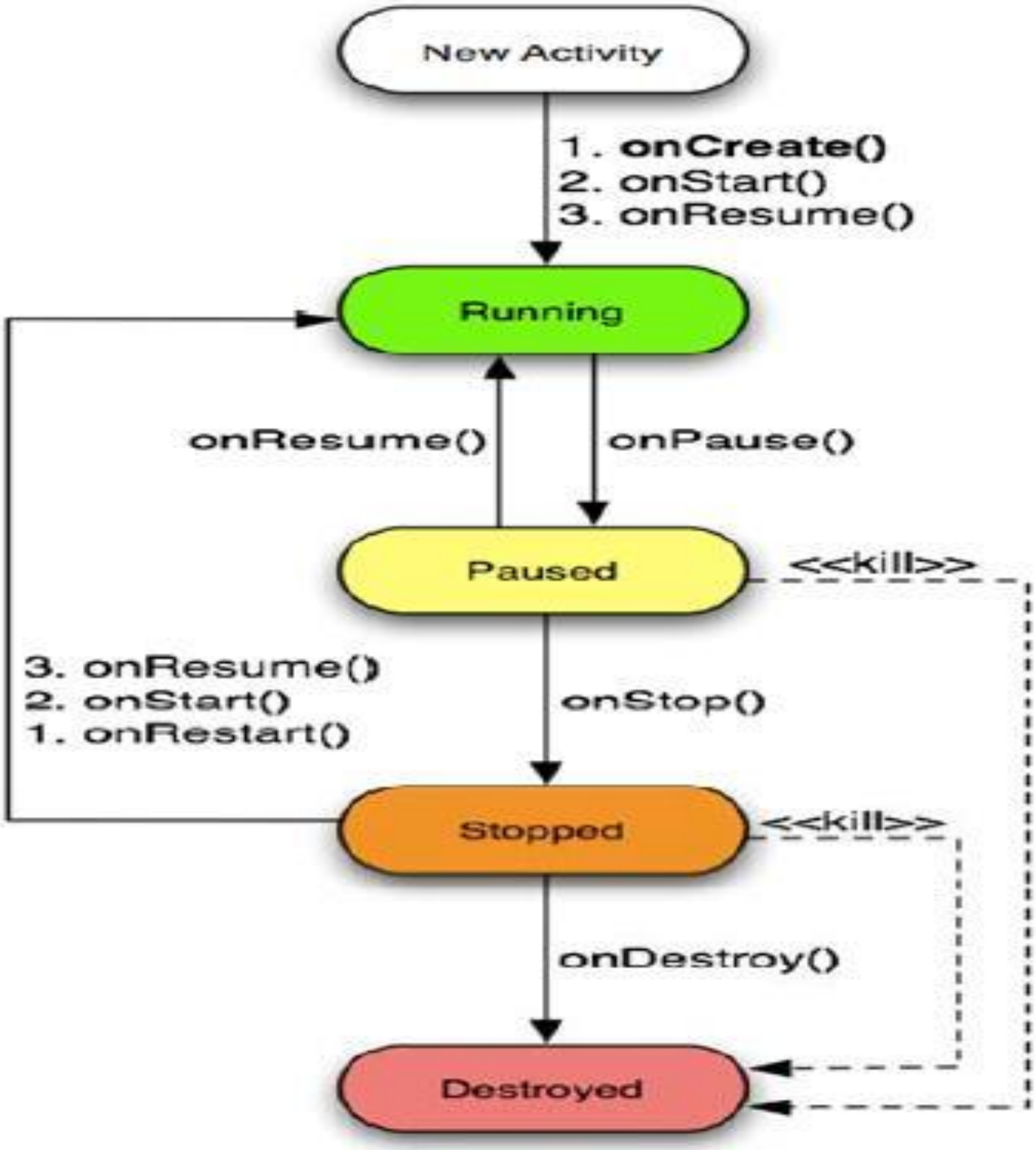
1. It is ***active* or *running***

2. It is ***paused*** or

3. It is ***stopped*** .


New Activity
1. onCreate()
2. onStart()
3. onResume()
Running
onResume()
onPause()
Paused
<<kill>>
3. onResume()
2. onStart()
1. onRestart()
onStop()
Stopped
<<kill>>
onDestroy()
Destroyed


New Activity
1. onCreate()
2. onStart()
3. onResume()
Running
onResume()
onPause()
Paused
<<kill>>
3. onResume()
2. onStart()
1. onRestart()
onStop()
Stopped
<<kill>>
onDestroy()
Destroyed


New Activity
1. onCreate()
2. onStart()
3. onResume()
Running
onResume()
onPause()
Paused
<<kill>>
3. onResume()
2. onStart()
1. onRestart()
onStop()
Stopped
<<kill>>
onDestroy()
Destroyed

An activity has essentially three states:

1. It is ***active* or *running*** when it is in the *foreground* of the screen(at the top of the *activity stack* for the current task).This is the activity that is the focus for the user's actions.

2. It is ***paused*** if it has lost focus but is still visible to the user.That is, another activity lies on top of it and that new activity either is*transparent* or *doesn't cover the full screen*. A paused activity is completely *alive* (it maintains all state and memberinformation and remains attached to the window manager), but can bekilled by the system in extreme low memory situations.

3. It is ***stopped*** if it is completely *obscured* by another activity.It still retains all state and member information. However*, it is no longervisible* to the user so its window is hidden and it will often be killed bythe system when memory is needed elsewhere

Life Cycle Events

If an activity is paused or stopped, the system can drop it from memoryeither by asking it to finish (calling its **finish**() method), or simply killing itsprocess.When it is displayed again to the user, it must be completely restarted andrestored to its previous state.As an activity transitions from state to state, it is notified of the change bycalls to the following protected *transition* methods:

All of these methods are **hooks** that you can override to do appropriatework when the state changes.All activities must implement **onCreate()** to do the initial setup when the object is first instantiated.Many activities will also implement **onPause()** to commit data changes andotherwise prepare to stop interacting with the user.The seven transition methods define the entire lifecycle of an activity.The **entire lifetime** of an activity happens between the first call to onCreate() through to a single final call to onDestroy().An activity does all its initial setup of "global" state in onCreate(), andreleases all remaining resources in onDestroy().

Visible Lifetime

The **visible lifetime** of an activity happens between a call to onStart() until acorresponding call to onStop().During this time, the user can see the activity on-screen, though it may notbe in the foreground and interacting with the user.The onStart() and onStop() methods can be called multiple times, as theactivity alternates between being visible and hidden to the user.Between these two methods, you can maintain resources that are neededto show the activity to the user.

Foreground Lifetime

The **foreground lifetime** of an activity happens between a call toonResume() until a corresponding call to onPause().During this time, the activity is in front of all other activities on screen and isinteracting with the user.An activity can frequently transition between the *resumed* and *paused*states — for example,

- *onPause*() is called when the device goes to sleep or when a newactivity is started,
- *onResume*() is called when an activity result or a new intent isdelivered.

Killable States

- Activities on killable states can be terminated by the system *at anytime after the method returns, without executing another line of theactivity's code*.
- Three methods (*onPause*(), *onStop*(), and *onDestroy*()) are *killable.*
- *onPause*() is the only one that is guaranteed to be called before theprocess is killed — *onStop*() and *onDestroy*() may not be.
- Therefore, you should use *onPause*() to write any persistent data (such as user edits) to storage

ANDROID DEVELOPMENT TOOLS

What are the Android Development Tools?

- Google provides the Android Development Tools (ADT) to develop Android applications with Eclipse. ADT is a set of plug-in which extended the Eclipse IDE with Android development capabilities.
- ADT contains all required functionality to create, compile, debug and deploy Android applications from the Eclipse IDE and from the command line. Other IDE's, e.g. IntellJ, are also reusing components of ADT.
- ADT also provides an Android device emulator, so that Android applications can be tested without a real Android phone.

How to develop Android Applications

- Android applications are primary written in the Java programming language. The Java source files are converted to Java class files by the Java compiler.
- Android provides a tool dx which converts Java class files into a dex (Dalvik Executable) file. All class files of one application are placed in one compressed .dex file. During this conversion process redundant information in the class files are optimized in the .dex file.

For example if the same String in different class file is found, the .dex file is stored only once and reference this String in the corresponding classes.

- .dex files are therefore much smaller in size then the corresponding class files.

- The .dex file and the resources of an Android project, e.g. the images and XML files are packed into an .apk(Android Package) file. The program aapt (Android Asset Packaging Tool) perform this packaging.
- The resulting .apk file contains all necessary data to run the Android application and can be deployed to an Android device via the "adb" tool.
- The Android Development Tools (ADT) allows that all these steps are performed transparent to the user; either within Eclipse or via the command line.
- If you use the ADT tooling you press a button or run a script and the whole Android application (.apk file) will be created and deployed.

ANDROID VIRTUAL DEVICE - EMULATOR

What is the Android Emulator?

- The Android Development Tools (ADT) include an emulator to run an Android system. The emulator behaves like a real Android device (in most cases) and allows you to test your application without having a real device.

  You can configure the version of the Android system you would like to run, the size of the SD card, the screen resolution and other relevant settings. You can define several devices with different configurations.

- Via the emulator you select which device should be started, you can also start several in parallel. These devices are called "Android Virtual Device" (AVD).

- The ADT allow to deploy and run your Android program on the AVD.

Emulator Shortcuts

Obviously you can use the emulator via the keyboard on the right side of the emulator. But there are also some nice shortcuts which are useful.

- **Alt**+**Enter** Maximizes the emulator. Nice for demos.
-

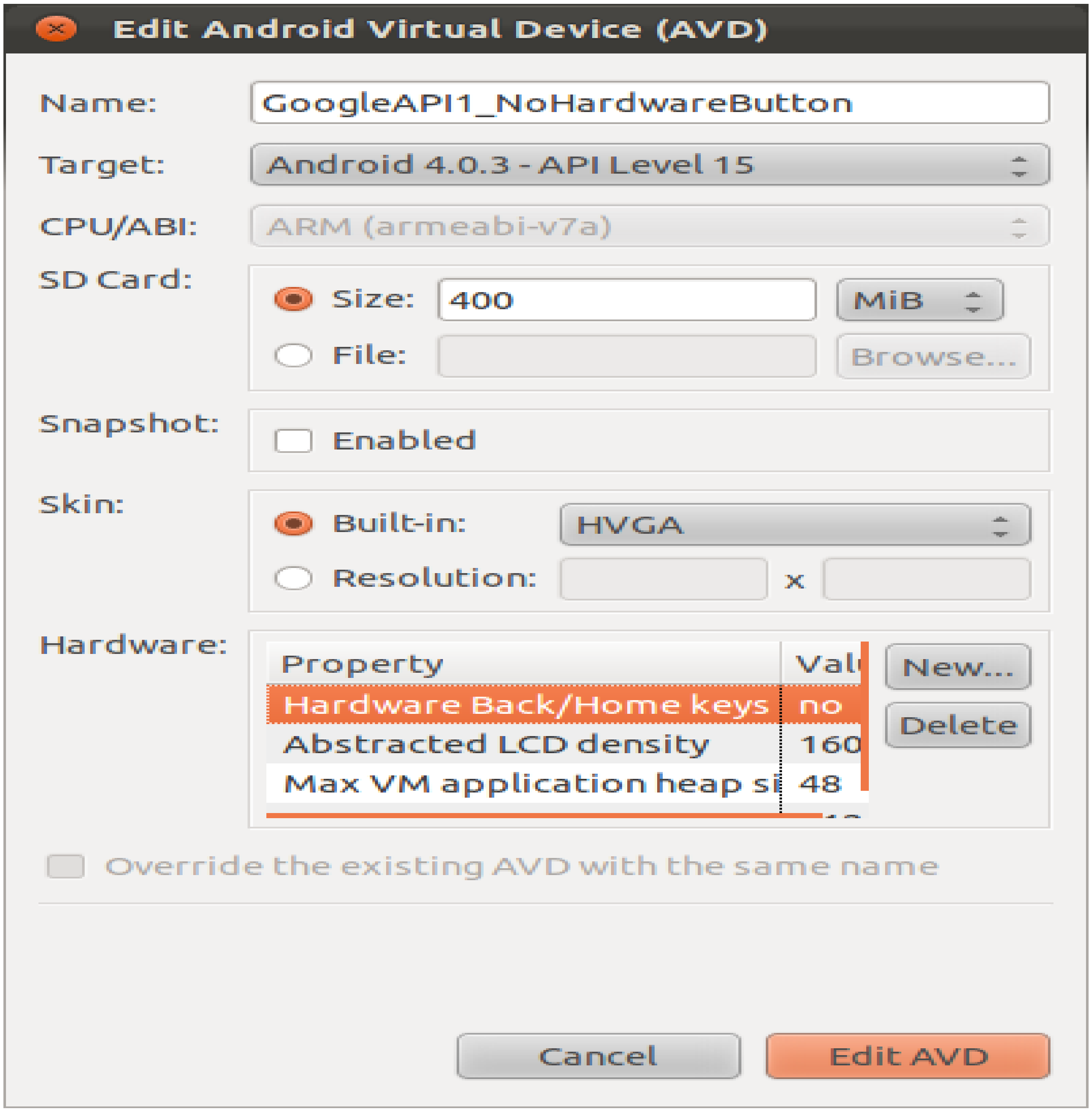
- **Ctrl+F11** changes the orientation of the emulator.
- **F8** Turns network on / off.

Performance

- Try to use a smaller resolution for your emulator as for example HVGA. The emulator gets slower the more pixels its needs to render as it is using software rendering.
- Also if you have sufficient memory on your computer, add at least 1 GB of memory to your emulator. This is the value "Device ram size" during the creation of the AVD.
- Also set the flag "Enabled" for Snapshots. This will save the state of the emulator and let it start much faster.

Hardware button

Android 4.0 introduced that devices do not have to have hardware button anymore. If you want to create such an AVD, add the "Hardware Back/Home keys" property to the device configuration and set it to "false".



Edit Android Virtual Device (AVD)

Name: GoogleAPI1_NoHardwareButton

Target: Android 4.0.3 - API Level 15

CPU/ABI: ARM (armeabi-v7a)

SD Card: [✓ Size:] 400 MiB File: Browse...

Snapshot: Enabled

Skin: [✓ Built-in:] HVGA Resolution: x

Hardware:

Property	Val
Hardware Back/Home keys	no
Abstracted LCD density	160
Max VM application heap si	48

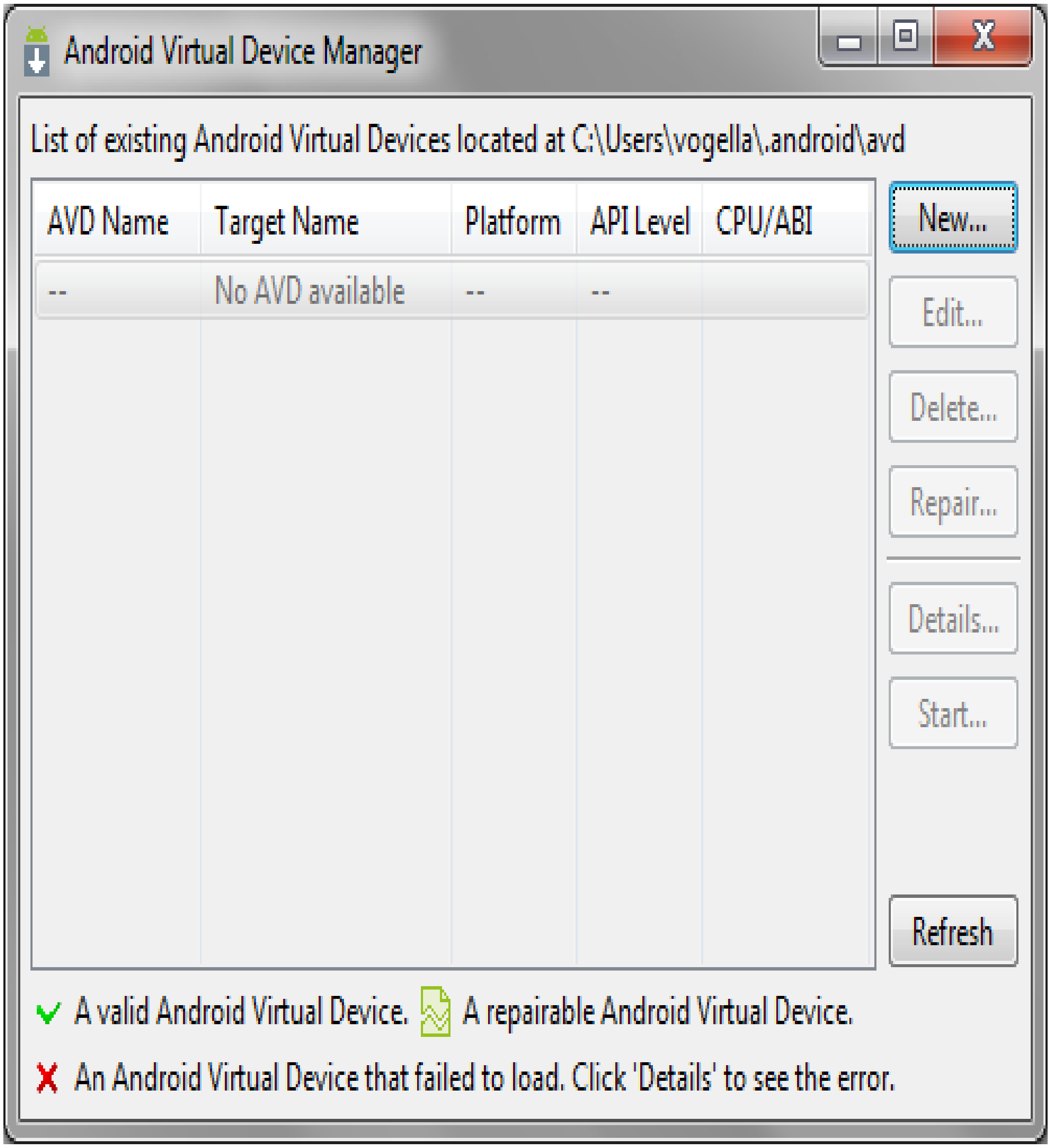
New... Delete

Override the existing AVD with the same name

Cancel Edit AVD

CREATE AND RUN ANDROID VIRTUAL DEVICE

To define an Android Virtual Device (ADV) open the "AVD Manager" via Windows → AVD Manager and press "New".

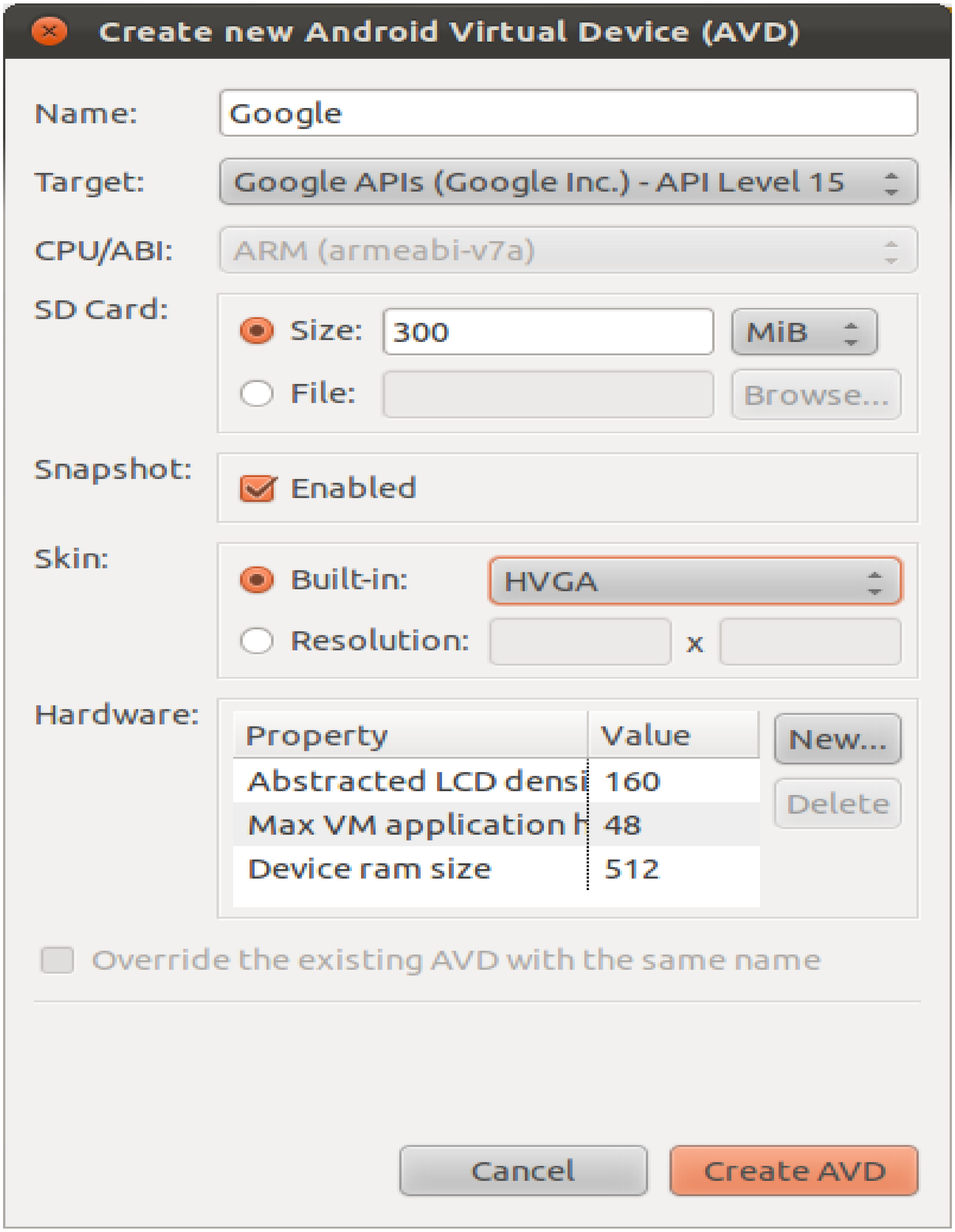


List of existing Android Virtual Devices located at C:\Users\vogella\.android\avd

AVD Name	Target Name	Platform	API Level	CPU/ABI
--	No AVD available	--	--	

✓ A valid Android Virtual Device. A repairable Android Virtual Device.

✗ An Android Virtual Device that failed to load. Click 'Details' to see the error.



Create new Android Virtual Device (AVD)

Name: Google

Target: Google APIs (Google Inc.) - API Level 15

CPU/ABI: ARM (armeabi-v7a)

SD Card: (● Size: 300 MiB) ○ File: Browse...

Snapshot: ✓ Enabled

Skin: (● Built-in: HVGA) ○ Resolution: x

Hardware:

Property	Value
Abstracted LCD densi	160
Max VM application h	48
Device ram size	512

New... Delete

☐ Override the existing AVD with the same name

Cancel Create AVD

Enter the following.

We can also select the box "Enabled" for Snapshots. This will make the second start of the virtual device much faster.

At the end press the button "Create AVD". This will create the AVD configuration and display it under the "Virtual devices".

To test if your setup is correct, select your device and press "Start".

After (a long time) your AVD starts. You are able to use it via the mouse and via the virtual keyboard of the emulator.

- During deployment on an Android device, the Android system will create a unique user and group ID for every Android application. Each application file is private to this generated user, e.g. other applications cannot access these files.
- In addition each Android application will be started in its own process.
- Therefore by means of the underlying Linux operating system, every Android application is isolated from other running applications. A misbehaving application cannot easily harm other Android applications.
- If data should be shared the application must do this explicitly, e.g. via a Service or a ContentProvider.
- Android also contains a permission system. Android predefines permissions for certain tasks but every application can also define its own permissions.

- An application must declare in its configuration file (AndroidManifest.xml) that it requires certain permissions.
- Depending on the details of the defined permission, the Android system will during installation either automatically grant the permission, reject it or ask the user if he grants these permissions to the application.

5.2. SECURITY & PERMISSIONS

- If for example the application declares that is requires Internet access then the user need to confirm this during installation.
- This is called "user driven security". The user decides to grant a permission or to deny it. If the user does not want to give all permissions required by the application, this application cannot be installed. The check of the permission is only performed during installation; permissions cannot be denied or granted after the installation.
- Typically not all users check the permissions in detail but some users do and if there is something strange with them, they will write bad reviews on the corresponding Android markets.

CHAPTER 6

SYSTEM TESTING

6.1 OVERVIEW

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing also provides an objective, independent view of the software to allow the business to appreciate and understand the risks at implementation of the software. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs.

Software testing can also be stated as the process of validating and verifying that a software program/application/product:

- meets the business and technical requirements that guided its design and development;
- works as expected; and
- Can be implemented with the same characteristics.

Software testing, depending on the testing method employed, can be implemented at any time in the development process. However, most of the test effort occurs after the requirements have been defined and the coding process has been completed. As such, the methodology of the test is governed by the software development methodology adopted.

Different software development models will focus the test effort at different points in the development process. Newer development models, such as Agile, often employ test driven development and place an increased portion of the testing in the hands of the developer, before it reaches a formal team of testers. In a more traditional model, most of the test execution occurs after the requirements have been defined and the coding process has been completed.

Testing can never completely identify all the defects within software. Instead, it furnishes a criticism or comparison that compares the state and behavior of the product against oracles—principles or mechanisms by which someone might recognize a problem. These oracles may include (but are not limited to) specifications, contracts, comparable products, past versions of

the same product, inferences about intended or expected purpose, user or customer expectations, relevant standards, applicable laws, or other criteria.

Every software product has a target audience. For example, the audience for video game software is completely different from banking software. Therefore, when an organization develops or otherwise invests in a software product, it can assess whether the software product will be acceptable to its end users, its target audience, its purchasers, and other stakeholders. Software testing is the process of attempting to make this assessment.

Functional vs. Non-functional testing

Functional testing refers to tests that verify a specific action or function of the code. These are usually found in the code requirements documentation, although some development methodologies work from use cases or user stories. Functional tests tend to answer the question of "can the user do this" or "does this particular feature work".

Non-functional testing refers to aspects of the software that may not be related to a specific function or user action, such as scalability or security. Non-functional testing tends to answer such questions as "how many people can log in at once", or "how easy is it to hack this software".

Defects and Failures

Not all software defects are caused by coding errors. One common source of expensive defects is caused by requirement gaps, e.g., unrecognized requirements, that result in errors of omission by the program designer. A common source of requirements gaps is non-functional requirements such as testability, scalability, maintainability, usability, performance, and security.

Software faults occur through the following processes. A programmer makes an error (mistake), which results in a defect (fault, bug) in the software source code. If this defect is executed, in certain situations the system will produce wrong results, causing a failure. Not all defects will necessarily result in failures. For example, defects in dead code will never result in failures. A defect can turn into a failure when the environment is changed. Examples of these changes in environment include the software being run on a new hardware platform, alterations in source data or interacting with different software. A single defect may result in a wide range of failure symptoms.

Static vs. dynamic testing

There are many approaches to software testing. Reviews, walkthroughs, or inspections are considered as static testing, whereas actually executing programmed code with a given set of test cases is referred to as dynamic testing. Static testing can be (and unfortunately in practice often is) omitted. Dynamic testing takes place when the program itself is used for the first time (which is generally considered the beginning of the testing stage). Dynamic testing may begin before the program is 100% complete in order to test particular sections of code (modules or discrete functions). Typical techniques for this are either using stubs/drivers or execution from a debugger environment. For example, spreadsheet programs are, by their very nature, tested to a large extent interactively ("on the fly"), with results displayed immediately after each calculation or text manipulation.

.2 SOFTWARE VERIFICATION AND VALIDATION

Software testing is used in association withverification and validation:

- Verification: Have we built the software right? (i.e., does it match the specification).

- Validation: Have we built the right software? (i.e., is this what the customer wants).

Software Quality Assurance (SQA)

Though controversial, software testing may be viewed as an important part of the software quality assurance (SQA) process. In SQA, software process specialists and auditors take a broader view on software and its development. They examine and change the software engineering process itself to reduce the amount of faults that end up in the delivered software: the so-called *defect rate.*

What constitutes an "acceptable defect rate" depends on the nature of the software. For example, an arcade video game designed to *simulate* flying an airplane would presumably have a much higher tolerance for defects than mission critical software such as that used to control the functions of an airliner that *really is* flying!

Although there are close links with SQA, testing departments often exist independently, and there may be no SQA function in some companies.

6.3 TESTING METHODS

Software testing methods are traditionally divided into white- and black-box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

6.3.1 White Box Testing

White box testing is when the tester has access to the internal data structures and algorithms including the code that implement these.

Types of white box testing

The following types of white box testing exist:

- **API testing (application programming interface)** - testing of the application using public and private APIs
- **Code coverage** - creating tests to satisfy some criteria of code coverage (e.g., the test designer can create tests to cause all statements in the program to be executed at least once)
- **Fault injection methods** - improving the coverage of a test by introducing faults to test code paths
- **Static testing** - White box testing includes all static testing

6.3.2 Black Box Testing

Black box testing treats the software as a "black box"—without any knowledge of internal implementation. Black box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, fuzz testing, model-based testing, traceability matrix, exploratory testing and specification-based testing.

Advantages and disadvantages

The black box tester has no "bonds" with the code, and a tester's perception is very simple: a code *must* have bugs. Using the principle, "Ask and you shall receive," black box testers find bugs where programmers do not. *But,* on the other hand, black box testing has been said to be "like a walk in a dark labyrinth without a flashlight," because the tester doesn't know how the software being tested was actually constructed. As a result, there are situations when (1) a tester writes many test cases to check something that could have been tested by only one test case, and/or (2) some parts of the back-end are not tested at all.

Therefore, black box testing has the advantage of "an unaffiliated opinion," on the one hand, and the disadvantage of "blind exploring," on the other.

6.3.3Testing Levels

Tests are frequently grouped by where they are added in the software development process, or by the level of specificity of the test.

Unit testing

Unit testing refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors.

These type of tests are usually written by developers as they work on code (white-box style), to ensure that the specific function is working as expected. One function might have multiple tests, to catch corner cases or other branches in the code. Unit testing alone cannot verify the functionality of a piece of software, but rather is used to assure that the building blocks the software uses work independently of each other.

Unit testing is also called *component testing*.

Integration testing

Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Software components may be integrated in an iterative way or all together ("big bang"). Normally the former is considered a better practice since it allows interface issues to be localised more quickly and fixed.

Integration testing works to expose defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.

System testing

System testing tests a completely integrated system to verify that it meets its requirements.

Unit testing

Unit testing refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors.

These type of tests are usually written by developers as they work on code (white-box style), to ensure that the specific function is working as expected. One function might have multiple tests, to catch corner cases or other branches in the code. Unit testing alone cannot verify the functionality of a piece of software, but rather is used to assure that the building blocks the software uses work independently of each other.

Unit testing is also called *component testing*.

Integration testing

Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Software components may be integrated in an iterative way or all together ("big bang"). Normally the former is considered a better practice since it allows interface issues to be localised more quickly and fixed.

Integration testing works to expose defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.

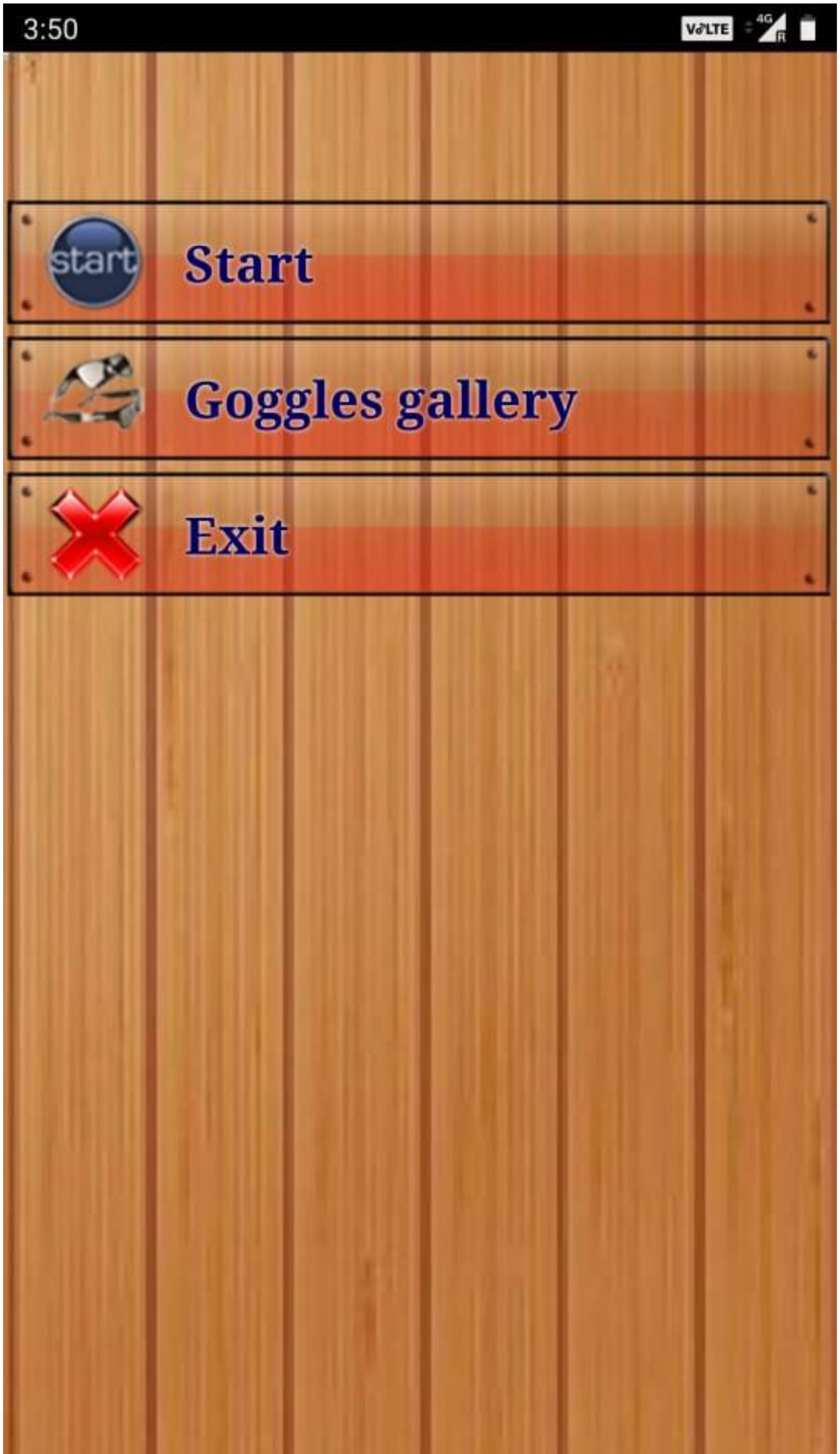
System testing

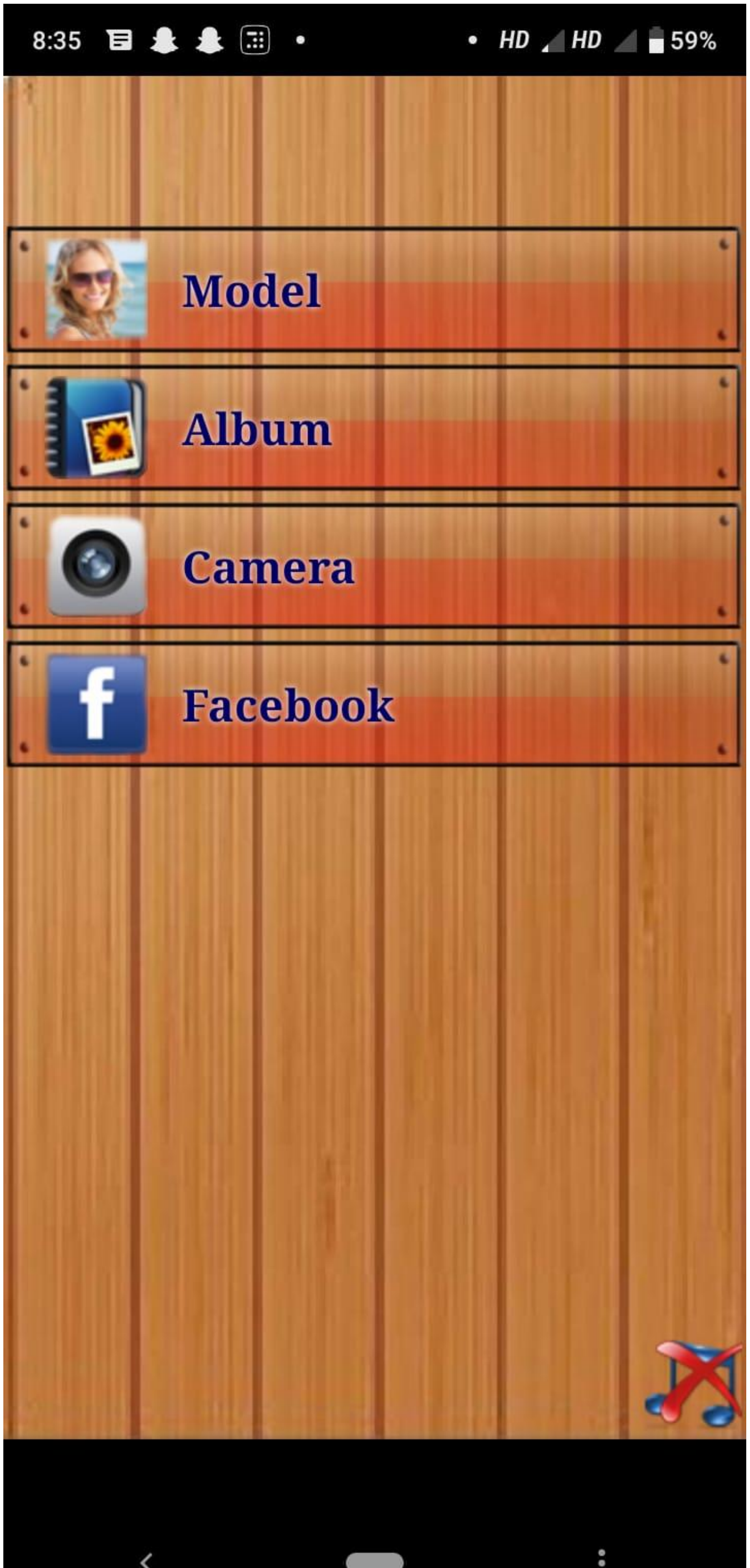
System testing tests a completely integrated system to verify that it meets its requirement

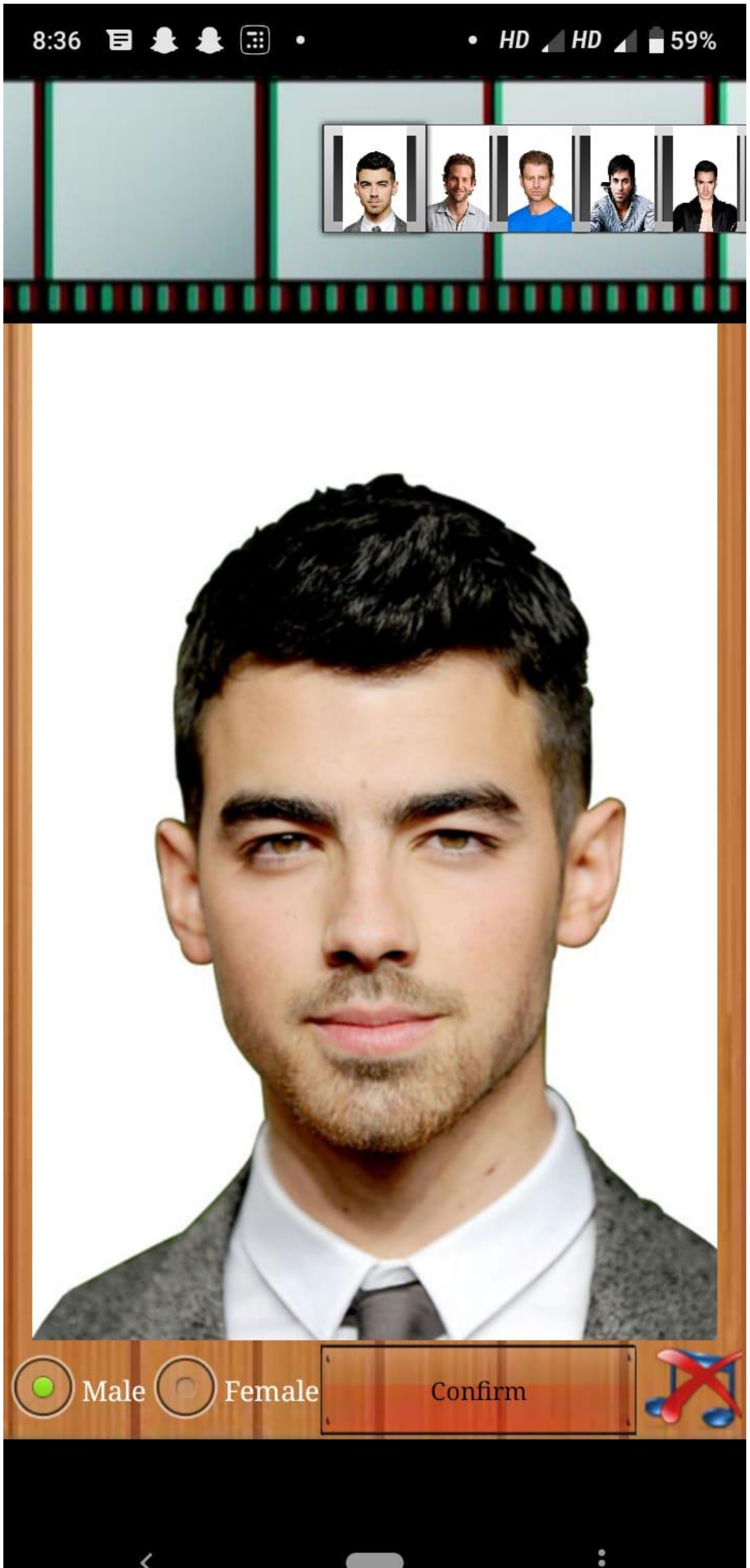
CHAPTER 8

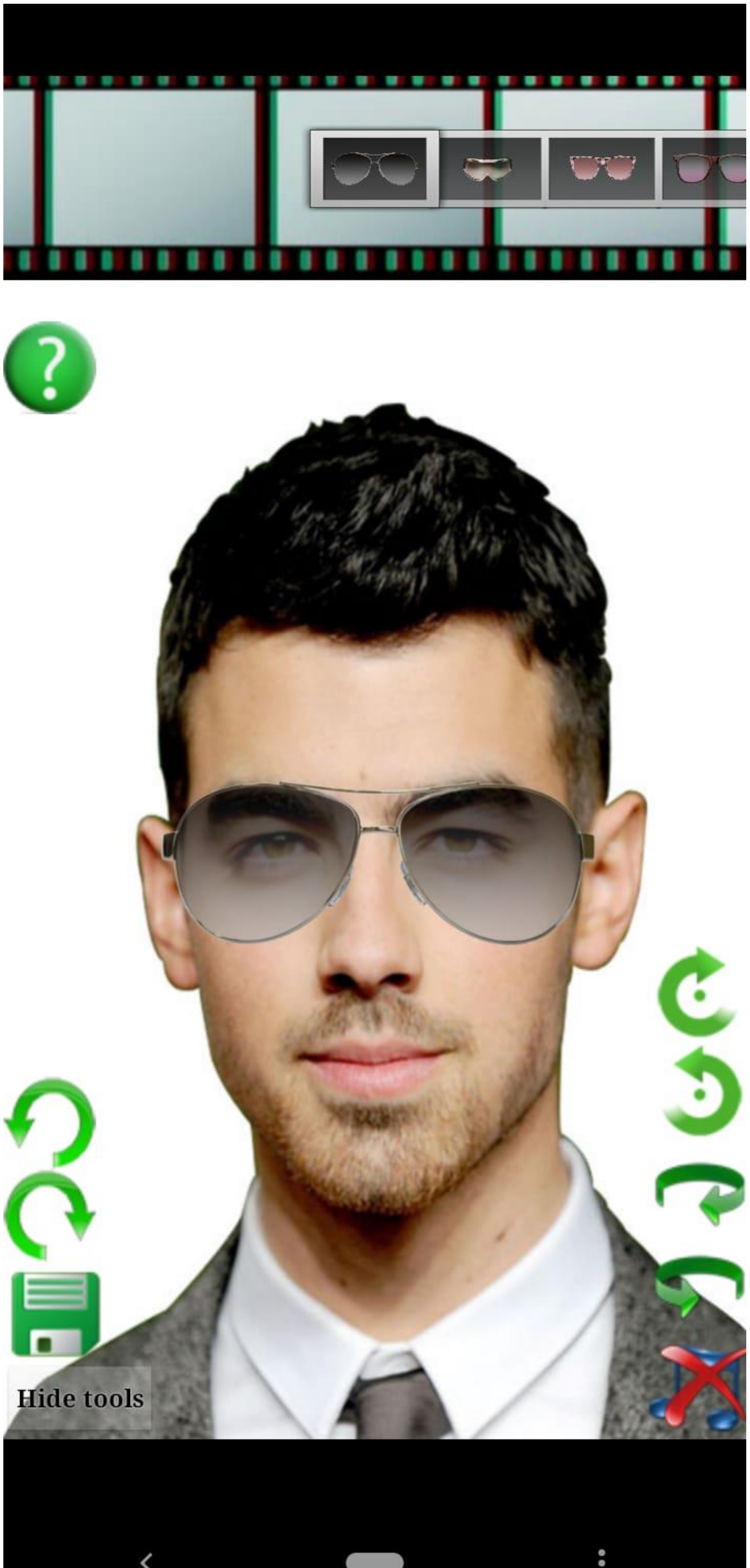
SCREEN LAYOUT

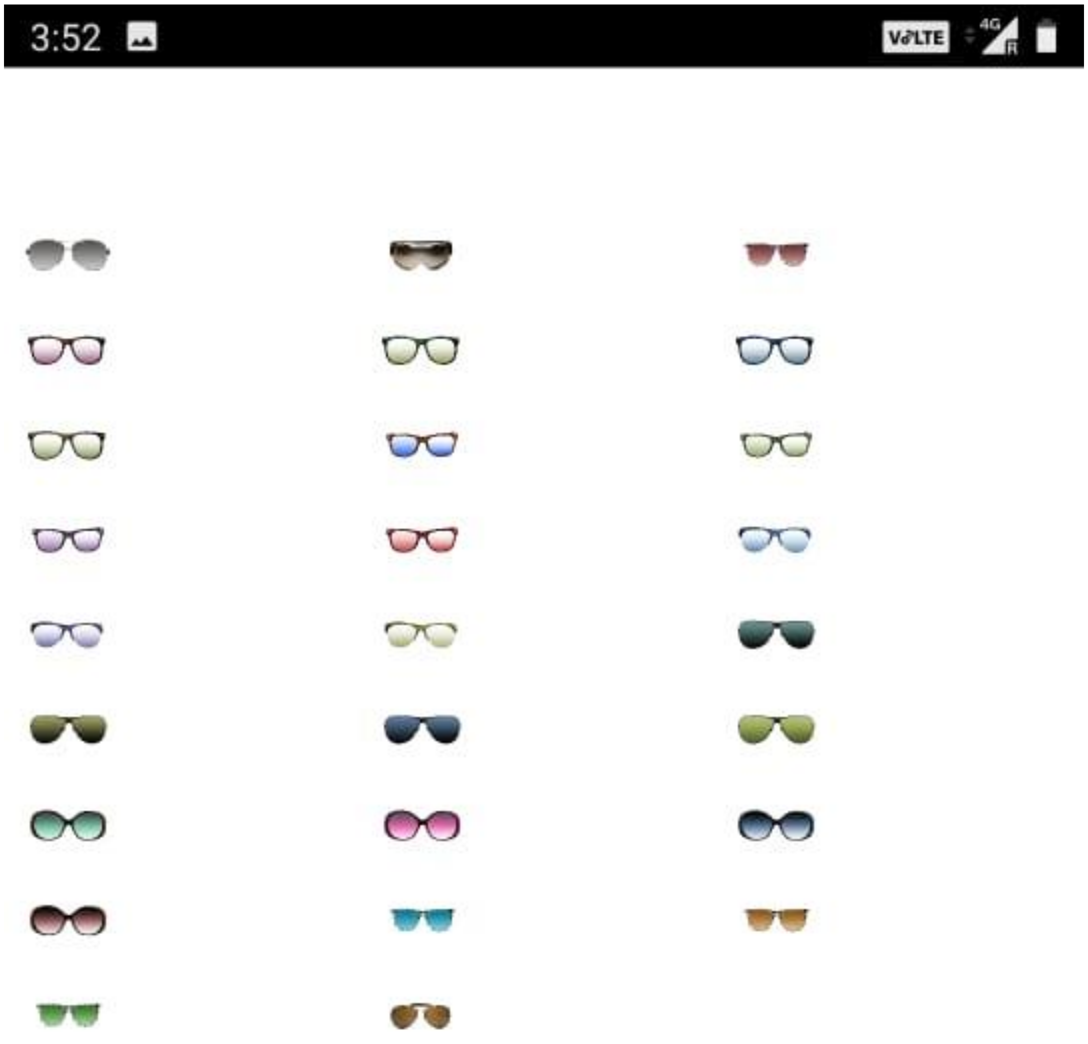
VIRTUAL
TRY ON


3:50
start
Start
Goggles gallery
Exit


8:35
HD
HD
59%
Model
Album
Camera
Facebook


8:36
HD HD 59%
Male
Female
Confirm


Hide tools


3:52
VoLTE
4G
R

CHAPTER 9

CODING

9.1 GRAPHICAL CODING

9.1.1.

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

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/layout" android:orientation="vertical"
android:layout_width="fill_parent"
        android:layout_height="fill_parent">
<TextView android:text="Welcome"
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="30dip"
        android:textStyle="bold"
        android:textColor="#ffffff"
        android:layout_marginLeft="85px">
</TextView>
<ImageView android:id="@+id/test_image"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:src="@drawable/logo1">
</ImageView>
```

```
<TextView android:text="Goggle combines most of the android phone apps in a single app as
well as it provides a direct interaction between the hiring session and the employer of the Goggle
organization via web services."

        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textStyle="bold"
        android:textSize="20px"
        android:textColor="#ffffff"
        android:paddingLeft="10px" android:layout_marginTop="8px">
</TextView>
<LinearLayout android:id="@+id/layout1"
        android:orientation="horizontal"
        android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="10px">
</LinearLayout>
<RelativeLayout android:id="@+id/RelativeLayout01"
   android:layout_height="wrap_content" android:layout_width="fill_parent">
<Button android:id="@+id/enter"
        android:layout_height="wrap_content"
        android:text="Enter"
```

```
            android:layout_width="100px"

            android:textStyle="bold"

            android:textSize="18px"

            android:textColor="#0066cc"

            android:layout_alignParentLeft="true"

            android:layout_alignParentBottom="true"


            android:layout_marginBottom="15px">

</Button>

<Button android:id="@+id/exit"

            android:layout_height="wrap_content"

            android:text="Exit"

            android:layout_width="100px"

            android:textSize="18px" android:textStyle="bold" android:textColor="#0066cc"

            android:layout_alignParentRight="true"

            android:layout_alignParentBottom="true"

            android:layout_marginBottom="15px">

</Button>

</RelativeLayout

</LinearLayout>
```

9.1.2.

```
<?xml version="1.1" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
            android:orientation="vertical"
            android:layout_height="fill_parent" android:layout_width="fill_parent">
<TextView android:id="@+id/name" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:layout_marginLeft="10dip"
android:layout_marginBottom="10dip" android:layout_marginTop="10dip"
android:textSize="16dip"></TextView>
<LinearLayout android:layout_height="wrap_content" android:orientation="horizontal"
android:layout_width="fill_parent">


<LinearLayout android:layout_height="wrap_content" android:layout_width="wrap_content"
android:orientation="vertical" android:layout_marginTop="10dip"
android:layout_marginBottom="10dip">
<TextView android:id="@+id/score" android:layout_height="wrap_content" android:text=""
android:textSize="20dip" android:layout_marginLeft="170dip"
android:layout_width="80dip"></TextView>
<TextView  android:layout_height="wrap_content" android:text="Match"
android:textSize="20dip" android:layout_marginLeft="170dip"
android:layout_width="80dip"></TextView>
</LinearLayout>
<LinearLayout android:layout_height="wrap_content" android:orientation="vertical"
android:layout_width="wrap_content" android:layout_marginBottom="4dip"
android:layout_marginLeft="5dip" android:visibility="gone">
```

```
<LinearLayout android:layout_height="wrap_content"  android:layout_width="wrap_content"
android:orientation="horizontal">
<View android:layout_width="120dip" android:background="#f00"
android:layout_height="15dip" android:layout_marginBottom="7dip"/>
<TextView android:id="@+id/Red" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:layout_marginLeft="4dip"></TextView>
</LinearLayout>
<LinearLayout android:layout_height="wrap_content" android:id="@+id/linearLayout9"
android:layout_width="wrap_content" android:orientation="horizontal">
<View android:layout_width="120dip" android:background="#0f0"
android:layout_height="15dip" android:layout_marginBottom="7dip"/>
<TextView android:id="@+id/Green" android:layout_width="wrap_content"

android:layout_height="wrap_content" android:layout_marginLeft="4dip"></TextView>
</LinearLayout>
<LinearLayout android:layout_height="wrap_content" android:id="@+id/linearLayout7"
android:layout_width="wrap_content" android:orientation="horizontal">
<View android:layout_width="120dip" android:background="#00f"
android:layout_height="15dip" android:layout_marginBottom="7dip"/>
<TextView android:id="@+id/Blue" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:layout_marginLeft="4dip"></TextView>
</LinearLayout>
```

```
<LinearLayout android:layout_height="wrap_content" android:id="@+id/linearLayout3"
android:layout_width="wrap_content" android:orientation="horizontal">
<View android:layout_width="120dip" android:background="#ffff00"
android:layout_height="15dip" />
<TextView android:id="@+id/Yellow" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:layout_marginLeft="4dip"></TextView>
</LinearLayout>
<TextView android:id="@+id/candidateId"  android:layout_width="wrap_content"
android:layout_height="wrap_content" android:visibility="gone"
android:layout_marginLeft="4dip"></TextView>
<TextView android:id="@+id/url"  android:layout_width="wrap_content"
android:layout_height="wrap_content" android:visibility="gone"></TextView>
</LinearLayout>
            </LinearLayout>
</LinearLayout>
```

9.1.3.

```
<?xml version="1.1" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:background="#ffff"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
```

```
<LinearLayout android:layout_height="wrap_content"
            android:orientation="horizontal" android:background="#8A9DB7"
            android:layout_width="fill_parent">
            <Button android:id="@+id/login_button" android:background="#486B9D"
                   android:textStyle="bold" android:layout_marginLeft="6dip"
                   android:layout_width="60dip" android:textSize="14dp"
                   android:textColor="#ffffff" android:layout_height="35dip"
                   android:layout_marginTop="8dip" android:text="Back">
            </Button>
</LinearLayout>
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dip"
    android:text="@string/planet_prompt"
  />
<Spinner
    android:id="@+id/spinner"


    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:prompt="@string/planet_prompt"
  />
```

```
<Button android:layout_gravity="center" android:id="@+id/login_button"
        android:text="Submit" android:textStyle="bold" android:layout_width="110dip"
        android:layout_height="50dip" android:textColor="#304d87"
        android:textSize="18dp" android:layout_marginTop="250px">
</Button>
</LinearLayout>
```

9.1.4.

```
<?xml version="1.1" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical"
        android:layout_width="fill_parent" android:layout_height="fill_parent">
        <LinearLayout
  android:layout_height="55dip"
                android:orientation="horizontal" android:background="#8A9DB7"
                android:layout_width="fill_parent">
                <Button android:id="@+id/button1" android:background="#486B9D"
                        android:textStyle="bold" android:layout_marginLeft="6dip"
                        android:layout_width="60dip" android:text="Back"
android:textSize="14dp"
                        android:textColor="#ffffff" android:layout_height="35dip"


                        android:layout_marginTop="8dip">
                </Button>
</LinearLayout>
        <LinearLayout android:orientation="vertical"
  android:layout_width="fill_parent"
                android:layout_height="fill_parent" >
        <TextView android:layout_width="wrap_content"
                android:layout_height="wrap_content" android:layout_marginTop="15dip"
android:layout_marginBottom="20dip"
                android:layout_gravity="center" android:text="Select Job From List"
android:textSize="18dip" />
        <Spinner android:id="@+id/spinner" android:layout_width="fill_parent"
                android:layout_height="wrap_content" android:prompt="@string/model" />
<Button
android:layout_height="42dip"
android:textSize="16dip"
android:textColor="#304d87"
android:layout_width="90dip"
        android:layout_gravity="center"
        android:layout_marginTop="20dip"
        android:textStyle="bold"
        android:id="@+id/send_email"
        android:text="Send">
</Button>
```

```
</LinearLayout>


</LinearLayout>
```

9.1.5.

```
<?xml version="1.1" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical"
        android:layout_height="fill_parent" android:layout_width="fill_parent">
        <LinearLayout  android:layout_height="50dip"
                android:orientation="horizontal" android:background="#8A9DB7"
                android:layout_width="fill_parent">
                <Button android:id="@+id/button1" android:background="#446B9D"
                        android:textStyle="bold" android:layout_marginLeft="6dip"
                        android:text="Applicants" android:textSize="14dp"
                        android:textColor="#ffffff" android:layout_height="35dip"
                        android:layout_marginTop="8dip" android:layout_width="85dip">
                </Button>
                <TextView android:id="@+id/score" android:text="" android:textColor="#ffffff"
                        android:textSize="22dip" android:textStyle="bold"
android:layout_marginTop="8dip"
                        android:layout_height="wrap_content" android:layout_width="162dip"
android:gravity="center"/>
        </LinearLayout>
```

```
            <LinearLayout android:orientation="vertical"
   android:layout_width="fill_parent"
                        android:layout_height="fill_parent">



            <Button android:id="@+id/postion_manager" android:textStyle="bold"
android:textColor="#304d87" android:textSize="15dp"
                        android:layout_width="200dip" android:layout_height="40dip"
                        android:text="To: Postion Manager" android:layout_marginTop="10dip"
android:layout_gravity="top|center">
            </Button>
            <Button android:id="@+id/other_manager" android:textStyle="bold"
android:textColor="#304d87" android:textSize="15dp"
                        android:layout_width="200dip" android:layout_height="40dip"
                        android:text="To: Other Manager" android:layout_marginTop="10dip"
android:layout_gravity="top|center">
            </Button>
            <Button android:id="@+id/individual" android:textStyle="bold"
android:textColor="#304d87" android:textSize="15dp"
                        android:layout_width="200dip" android:layout_height="40dip"
                        android:text="To: An Individual"  android:layout_marginTop="10dip"
android:layout_gravity="top|center">
            </Button>
            <Button android:id="@+id/to_me" android:textStyle="bold"
android:textColor="#304d87" android:textSize="15dp"
                        android:layout_width="200dip" android:layout_height="40dip"
                        android:text="To: Me" android:layout_marginTop="10dip"
android:layout_gravity="top|center">
            </Button>

</LinearLayout>
</LinearLayout>
```

9.1.6.

```
<?xml version="1.1" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_height="wrap_content"
                        android:orientation="horizontal" android:background="#8A9DB7"
                        android:layout_width="fill_parent">
                        <Button android:id="@+id/button1" android:background="#486B9D"
                                    android:textStyle="bold" android:layout_marginLeft="6dip"
                                    android:textSize="14dp"
                                    android:textColor="#ffffff" android:layout_height="35dip"
                                    android:layout_marginTop="8dip" android:text="Back"
android:layout_width="65dip">
```

```
                    </Button>
                    <TextView android:text="Applicants" android:textColor="#ffffff"
                              android:textSize="22dp" android:textStyle="bold"
android:layout_marginTop="8dip"
                              android:layout_height="43dip" android:layout_width="wrap_content"
android:layout_marginLeft="50dip"/>
</LinearLayout>
```

9.1.7.

```
<?xml version="1.1" encoding="utf-8"?>
<LinearLayout

  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_height="wrap_content"
                    android:orientation="horizontal" android:background="#8A9DB7"
                    android:layout_width="fill_parent">
                    <Button android:id="@+id/button1" android:background="#486B9D"
                              android:textStyle="bold" android:layout_marginLeft="6dip"
                              android:layout_width="60dip" android:text="Back"
android:textSize="14dp"
                              android:textColor="#ffffff" android:layout_height="35dip"
                              android:layout_marginTop="8dip">
                    </Button>
                    <TextView android:textColor="#ffffff"
                              android:textSize="22dp" android:layout_width="fill_parent"
                              android:textStyle="bold" android:layout_marginTop="8dip"
                              android:layout_height="43dip" android:layout_marginLeft="4dip"
android:text="Close Hiring Session"/>
</LinearLayout>
```

9.1.8.

```
<?xml version="1.1" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_height="wrap_content"
                    android:orientation="horizontal" android:background="#8A9DB7"
                    android:layout_width="fill_parent">

                    <Button android:id="@+id/button1" android:background="#486B9D"
                              android:textStyle="bold" android:layout_marginLeft="6dip"
                              android:layout_width="60dip" android:text="Back"
android:textSize="14dp"
                              android:textColor="#ffffff" android:layout_height="35dip"
                              android:layout_marginTop="8dip">
                    </Button>
          <TextView android:text="Email Job Posting" android:textColor="#ffffff"
                              android:textSize="22dp" android:layout_width="fill_parent"
                              android:textStyle="bold" android:layout_marginTop="8dip"
                              android:layout_height="43dip" android:layout_marginLeft="4dip"/>
</LinearLayout>
```

9.1.9.

```
<?xml version="1.1" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical" android:layout_width="fill_parent"
        android:layout_height="fill_parent">
<LinearLayout android:layout_height="wrap_content"
                android:orientation="horizontal" android:background="#8A9DB7"
                android:layout_width="fill_parent">
                <Button android:id="@+id/button1" android:background="#486B9D"
                    android:textStyle="bold" android:layout_marginLeft="6dip"
                    android:layout_width="60dip" android:text="Back"
android:textSize="14dp"
                    android:textColor="#ffffff" android:layout_height="35dip"

                    android:layout_marginTop="8dip">
                </Button>
                <TextView android:text="Model Profiles" android:textColor="#ffffff"
                    android:layout_width="fill_parent"
```

```
                              android:textStyle="bold" android:layout_height="43dip"
android:textSize="22dip" android:layout_marginLeft="25dip"
android:layout_marginTop="10dip"/>
</LinearLayout>
</LinearLayout>
```

9.1.10.

```
<?xml version="1.1" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_height="50dip"
                        android:orientation="horizontal" android:background="#8A9DB7"
                        android:layout_width="fill_parent">
                        <Button android:id="@+id/button1" android:background="#486B9D"
                              android:textStyle="bold" android:layout_marginLeft="6dip"
                              android:text="Back" android:textSize="14dp"
                              android:textColor="#ffffff" android:layout_height="35dip"
                              android:layout_marginTop="8dip" android:layout_width="70dip">
                        </Button>
                        <TextView android:id="@+id/heading" android:text=""
android:textColor="#ffffff"
                              android:textSize="22dip" android:textStyle="bold"

android:layout_marginTop="8dip"
```

```
                              android:layout_height="wrap_content" android:layout_width="240dip"
android:gravity="center"/>
</LinearLayout>
```

9.1.11.

```
<?xml version="1.1" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
            android:orientation="vertical"
            android:layout_width="fill_parent"  android:layout_height="fill_parent">
            <LinearLayout
  android:layout_height="50dip"
            android:orientation="horizontal" android:background="#8A9DB7"
            android:layout_width="fill_parent">
            <Button android:id="@+id/button1" android:background="#486B9D"
                  android:textStyle="bold" android:layout_marginLeft="6dip"
                  android:text="Back" android:textSize="14dp"
                  android:textColor="#ffffff" android:layout_height="35dip"
                  android:layout_marginTop="8dip" android:layout_width="70dip">
            </Button>
                        <TextView android:id="@+id/heading" android:text=""
android:textColor="#ffffff"
```

```
                              android:textSize="22dip" android:textStyle="bold"
android:layout_marginTop="8dip"


                              android:layout_height="wrap_content" android:layout_width="162dip"
android:gravity="center"/>
</LinearLayout>
                              <LinearLayout android:orientation="horizontal"
android:background="#ffffff"
                    android:layout_height="wrap_content" android:layout_width="fill_parent">
                    <TextView  android:textColor="#486B9D" android:text="Traits"
android:layout_height="wrap_content" android:layout_width="100dip"
                              android:textSize="20dip" android:layout_marginLeft="10dip"
android:id="@+id/con" android:gravity="center"></TextView>
                              <TextView android:textColor="#486B9D" android:text=""
android:layout_height="wrap_content" android:layout_width="100dip"
                              android:textSize="20dip" android:layout_marginLeft="95dip"
android:id="@+id/candidate" android:gravity="center"></TextView>
          </LinearLayout>
</LinearLayout>
```

9.1.12.

```
<?xml version="1.1" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
          android:orientation="vertical"
```

```
            android:layout_height="fill_parent"
            android:layout_width="fill_parent"
            >
            <ImageView
   android:id="@+id/test_image"


   <ListView
    android:id="@+id/android:list"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    />
<TextView
    android:id="@+id/android:empty"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:text="@string/main_no_items"/>
</LinearLayout>
```

9.1.15.

```
<?xml version="1.1" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
            android:orientation="vertical"
            android:layout_height="fill_parent" android:layout_width="fill_parent">
<LinearLayout
```

```
  android:orientation="horizontal" android:background="#8A9DB7"
                    android:layout_width="fill_parent" android:layout_height="50dip"
android:layout_marginBottom="18dip">
                    <Button android:id="@+id/close_app" android:background="#486B9D"
                              android:textStyle="bold" android:layout_marginLeft="6dip"
                              android:text="Close App" android:textSize="14dp"
                              android:textColor="#ffffff" android:layout_height="35dip"



                              android:layout_marginTop="8dip" android:layout_width="120dip">
                    </Button>
                    <Button android:id="@+id/sign_out" android:background="#486B9D"
                              android:textStyle="bold" android:text="Sign Out"
android:textSize="14dp"
                              android:textColor="#ffffff" android:layout_height="35dip"
                              android:layout_marginTop="8dip" android:layout_width="120dip"
android:layout_marginLeft="65dip">
                    </Button>
</LinearLayout>
          <ScrollView android:id="@+id/ScrollView01"
android:layout_width="fill_parent" android:layout_height="fill_parent">
          <LinearLayout android:orientation="vertical"
  android:layout_width="fill_parent"
```

```
                    android:layout_height="fill_parent">
<Button android:id="@+id/button1" android:textStyle="bold" android:textColor="#304d87"
android:textSize="18dip"
                    android:layout_width="270dip" android:text="Open Hiring Sessions"
android:layout_gravity="top|center" android:layout_height="45dip"
android:layout_marginBottom="18dip">
          </Button>
          <Button android:id="@+id/button2" android:textStyle="bold"
android:textColor="#304d87" android:textSize="18dip"
                    android:layout_width="270dip" android:layout_gravity="top|center"
android:text="Closed Hiring Sessions" android:layout_marginBottom="18dip"

android:layout_height="44dip">
          </Button>
          <Button android:id="@+id/button3" android:text="Create New Hiring Session"
                     android:layout_width="270dip" android:textStyle="bold"
android:textColor="#304d87" android:textSize="18dip"
                    android:layout_gravity="top|center" android:layout_marginBottom="18dip"
android:layout_height="44dip">
          </Button>
          <Button android:id="@+id/button4" android:text="Available Model Profiles"
                    android:textStyle="bold" android:textColor="#304d87" android:textSize="18dip"
                    android:layout_width="270dip" android:layout_gravity="top|center"
android:layout_marginBottom="18dip" android:layout_height="44dip">
          </Button>
          <Button android:id="@+id/button5" android:text="Email a Job Posting"
                    android:textStyle="bold" android:textColor="#304d87" android:textSize="18dip"
                    android:layout_width="270dip" android:layout_gravity="top|center"
android:layout_height="44dip">
          </Button>
</LinearLayout>
</ScrollView>
</LinearLayout>
```

9.1.16.

```
<?xml version="1.1" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
          android:orientation="vertical"


          android:layout_height="fill_parent" android:layout_width="fill_parent" >
<LinearLayout android:orientation="horizontal"
                    android:layout_height="wrap_content" android:paddingTop="10dip"
android:layout_width="wrap_content">
                    <TextView android:layout_height="wrap_content"
android:layout_width="wrap_content"
```

```
                              android:layout_marginRight="5dip" android:textSize="18dp"
android:id="@+id/bottomtext1"></TextView>
                    </LinearLayout>
          <LinearLayout android:orientation="horizontal" android:layout_marginTop="9dip"
                    android:layout_height="wrap_content" android:layout_width="fill_parent" >
                    <TextView android:id="@+id/textView1" android:layout_height="wrap_content"
android:layout_marginRight="5dip" android:textSize="16dp" android:text="Created On:  "
android:layout_width="wrap_content"></TextView>
                    <TextView android:layout_height="wrap_content"
android:layout_width="wrap_content"
                              android:textSize="14dip" android:layout_marginLeft="16dip"
android:id="@+id/toptext"></TextView>
          </LinearLayout>
          <LinearLayout android:orientation="horizontal"
                    android:layout_height="wrap_content" android:layout_width="wrap_content">
                    <TextView android:id="@+id/textView1" android:layout_height="wrap_content"
android:layout_width="wrap_content"
                              android:layout_marginRight="5dip" android:textSize="16dp"
android:text="Session Id:  "></TextView>

                    <TextView android:layout_height="wrap_content"
android:layout_width="wrap_content"
```

```
                              android:textSize="14dip" android:layout_marginLeft="24dip"
android:id="@+id/session_id" ></TextView>
          </LinearLayout>
          <LinearLayout android:orientation="horizontal" android:paddingBottom="10dip"
android:layout_marginTop="9dip"
                    android:layout_height="wrap_content" android:layout_width="wrap_content"
android:layout_marginLeft="55dip">
                    <TextView android:id="@+id/textView1" android:layout_height="wrap_content"
android:layout_width="wrap_content"
                              android:layout_marginRight="5dip" android:textSize="16dp"
android:text="Total Applicant's:"></TextView>
                    <TextView android:layout_height="wrap_content"
android:layout_width="wrap_content"
                              android:textSize="14dip" android:layout_marginLeft="16dip"
android:id="@+id/bottomtext"></TextView>
          </LinearLayout>
</LinearLayout>
```

8.2. FUNCTIONAL CODING

8.2.1

```
package com.Goggle;

import java.net.HttpURLConnection;

import java.net.URL;

import android.app.Activity;

import android.app.AlertDialog;

import android.content.Context;

import android.content.DialogInterface;

import android.net.ConnectivityManager;

public class CheckConnection {

        public static int NOT_CONNECTED = 0;

        public static int CONNECTED = 1;

        public static int SERVICE_UNAVAILABLE = 3;

        public static int SERVICE_AVAILABLE = 4;

HttpURLConnection urlc;

        ConnectivityManager cm;

public  int isOnline(Activity context, String urlstr) {
```

```
                    if(urlstr.indexOf("?")!= -1){

                              urlstr = urlstr.substring(0,urlstr.indexOf("?"));


                         }

                         int res = NOT_CONNECTED;

                         try {

                              cm = (ConnectivityManager)

context.getSystemService(Context.CONNECTIVITY_SERVICE);

                         cm.getActiveNetworkInfo().isConnectedOrConnecting();

                              res = CONNECTED;

                              String url1= "http://www.google.co.in/";

                              URL url = new URL(url1);

                               urlc = (HttpURLConnection) url.openConnection();

                               urlc.setConnectTimeout(100000);

urlc.connect();

                              } catch (Exception e) {

                              e.printStackTrace();

                         showAlertDialog(context,"Please check your internel Connection");

                         res = NOT_CONNECTED;
```

```
                        }
                                if(res == CONNECTED){
                        try {
                        URL url = new URL(urlstr);



                                urlc = (HttpURLConnection) url.openConnection();
                                urlc.setConnectTimeout(50000);
                        urlc.connect();
                        if (urlc.getResponseCode() == 200) {
                                        res = SERVICE_AVAILABLE;
//showAlertDialog(context,"Web Service is running");
                                }
 else {
                                        showAlertDialog(context,"Web Service is unavailable.
Please check after some time.");
                                        return SERVICE_UNAVAILABLE;
                                }
                        }
 catch (Exception e) {
                                e.printStackTrace();
                                showAlertDialog(context,"Web Service is unavailable. Please
check after some time.");
                                res = SERVICE_UNAVAILABLE;
                        }
                        }

                return res;
}
        public  void showAlertDialog(Activity context,String message){
                try {
                AlertDialog alertDialog = new AlertDialog.Builder(context ).create();
                  // alertDialog.setTitle("Alert 1");
                 alertDialog.setMessage(message);
                alertDialog.setButton("Cancel", new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int which) {
                      return;
                    } });
                    alertDialog.show();
                } catch (Exception e) {
                System.out.println("  Exception  "+e);
                }
        }
}
```

8.2.2

```
package com.Goggle;
import java.util.ArrayList;
import org.ksoap2.SoapEnvelope;
import org.ksoap2.serialization.SoapObject;
import org.ksoap2.serialization.SoapSerializationEnvelope;
import org.ksoap2.transport.AndroidHttpTransport;
import android.app.ListActivity;

import android.app.ProgressDialog;

import android.content.Context;

import android.content.Intent;

import android.os.Bundle;

import android.util.Log;

import android.view.KeyEvent;

import android.view.LayoutInflater;

import android.view.View;

import android.view.View.OnClickListener;

import android.view.ViewGroup;

import android.widget.AdapterView;

import android.widget.AdapterView.OnItemClickListener;



import android.widget.ArrayAdapter;

import android.widget.Button;

import android.widget.ListView;

import android.widget.TextView;
```

```
public class Close extends ListActivity implements OnClickListener

{

private ProgressDialog m_ProgressDialog = null;

        private ArrayList<Order1> m_orders = null;

        private OrderAdapter m_adapter;

        private Runnable viewOrders;

  String user_name;

@Override

        public void onCreate(Bundle savedInstanceState) {

                super.onCreate(savedInstanceState);

        Intent myIntent = getIntent();

          user_name = myIntent.getStringExtra("name");

                setContentView(R.layout.main);

                m_orders = new ArrayList<Order1>();

                this.m_adapter = new OrderAdapter(this, R.layout.close_hiring, m_orders);

        ListView lv = getListView();




         lv.setTextFilterEnabled(true);
```

```
                        LayoutInflater infalter = getLayoutInflater();
  ViewGroup header = (ViewGroup) infalter.inflate(R.layout.header_close, lv, false);
  lv.addHeaderView(header);
   Button button1 = (Button)findViewById(R.id.button1);
   button1.setOnClickListener(this);
       (this.m_adapter);
                        viewOrders = new Runnable() {
                                public void run() {
                                        getOrders();
                                }
                        };
                        Thread thread = new Thread(null, viewOrders, "MagentoBackground");
                        thread.start();
                m_ProgressDialog = ProgressDialog.show(Close.this, "Please wait...", "Retrieving data
...", true);
                lv.setOnItemClickListener(new OnItemClickListener() {
                        public void onItemClick(AdapterView<?> parent, View view,int position, long
id) {
                        Intent myIntent = new Intent(view.getContext(), Sessioncandidate.class);
                                TextView et = (TextView) view.findViewById(R.id.session_id);
```

```
                              System.out.println("***setting value to "+et.getText());

                              myIntent.putExtra("session_id", Integer.valueOf(et.getText().toString()));

myIntent.addFlags(myIntent.FLAG_ACTIVITY_FORWARD_RESULT);

                              startActivity(myIntent);

                              System.out.println("hhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh");

        }

                        });

      }

      private Runnable returnRes = new Runnable() {

                  public void run() {

                              if (m_orders != null && m_orders.size()

                                                      > 0) {

                                          m_adapter.notifyDataSetChanged();

                                          for (int i = 0; i < m_orders.size(); i++)

                                                      m_adapter.add(m_orders.get(i));

                              }

                              m_ProgressDialog.dismiss();

                              m_adapter.notifyDataSetChanged();
```

```
                }

            };




            private class OrderAdapter extends ArrayAdapter<Order1> {

                        private ArrayList<Order1> items;

                        public OrderAdapter(Context context, int textViewResourceId,
ArrayList<Order1> items) {

                                    super(context, textViewResourceId, items);

                                    this.items = items;

}                       @Override

                        public View getView(int position, View convertView, ViewGroup parent) {

                                    View v = convertView;

                                    if (v == null) {

                                                LayoutInflater vi = (LayoutInflater)
getSystemService(Context.LAYOUT_INFLATER_SERVICE);

                                                v = vi.inflate(R.layout.close_hiring, null);

                                    }

                                    Order1 o = items.get(position);

                                    if (o != null) {
```

```
                    TextView tt = (TextView) v.findViewById(R.id.toptext);

                    //tt.setTextSize(16);

 TextView bt = (TextView) v.findViewById(R.id.bottomtext);

                    TextView ct = (TextView) v.findViewById(R.id.bottomtext1);

                    TextView dt = (TextView) v.findViewById(R.id.bottomtext2);



                    TextView et = (TextView) v.findViewById(R.id.session_id);

if (tt != null) {

            tt.setText(o.getSession_name());

            }

                    if (bt != null) {

                        bt.setText(o.getSession_create());

                    }

                    if (ct != null) {

                        ct.setText(o.getClose_date());

                    }

 if (dt != null) {

                        dt.setText(o.getTotalApplicants());

                    }
 if (et != null) {
                     et.setText(o.getSession_id());
                            }
                     }
                     return v;
              }
}



private void getOrders() {
              try {
                     m_orders = new ArrayList<Order1>();
                     /*Order o1 = new Order();
                     o1.setOrderName("SF services");
                     o1.setOrderStatus("Pending");
                     Order o2 = new Order();
                     o2.setOrderName("SF Advertisement");
                     o2.setOrderStatus("Completed");
                     m_orders.add(o1);
```

```
                        m_orders.add(o2);*/
                        SoapObject Request = new SoapObject(NAMESPACE,
METHOD_NAME);
                        // SoapObject
                         Request.addProperty("status", 0);
                        Request.addProperty("username",user_name);
                        //Request.addProperty("passcode", "employer2");
                        // Request.addProperty("logging","YES");
                        //Request.addProperty("status", 1);
                        SoapSerializationEnvelope soapEnvelope = new
SoapSerializationEnvelope(SoapEnvelope.VER11);


                        soapEnvelope.dotNet = true;
                        soapEnvelope.setOutputSoapObject(Request);
                        //tv.setText("Status0 :");
                        AndroidHttpTransport abt = new AndroidHttpTransport(URL);
                        try {
                                abt.call(SOAP_ACTION, soapEnvelope);
                                SoapObject resultSOAP = (SoapObject)
soapEnvelope.getResponse();
```

```
                                /* gets our result in JSON String */
                                String ResultObject = resultSOAP.getProperty(0).toString();
                                SoapObject resultsRequestSOAP = (SoapObject)
soapEnvelope.bodyIn;
                                int elementCount = resultsRequestSOAP.getPropertyCount();
                                ArrayList<Order1> resultData = new ArrayList<Order1>();
                                if (elementCount > 0) {
                                        SoapObject element = (SoapObject) ((SoapObject)
resultsRequestSOAP.getProperty(0)).getProperty(1);
                                        int x = element.getPropertyCount();
                                        for (int i = 0; i < x; i++) {
                                                int props = element.getPropertyCount();
                                                int atrCnt = element.getAttributeCount();

                                                if (props > 0) {
                                                        SoapObject elementChild = (SoapObject)
element.getProperty(0);
                                                        int propsChild =
elementChild.getPropertyCount();
                                                        int atrCntChild =
elementChild.getAttributeCount();

                                                        for (int j = 0; j < propsChild; j++) {

                                                              SoapObject nodeElement =

(SoapObject) elementChild.getProperty(j);

                   m_orders.add(new

Order1(nodeElement.getProperty("TotalApplicants").toString(),

                               nodeElement.getProperty("session_name").toString(),

                               nodeElement.getProperty("session_id").toString(),

                               nodeElement.getProperty("session_create").toString(),

                               nodeElement.getProperty("close_date").toString() ));
                                                              }

                                                  }

                                            }

                                      }

                                      System.out.println(""+resultData.size());




                                      // SoapPrimitive resultString

                                      // =(SoapPrimitive)soapEnvelope.getResponse();

                                      // SoapObject resultString =
                    // ((SoapObject)soapEnvelope.getResponse());

                    // System.err.println(resultString.getName());

                    String s="";

                    for(Order1 order: resultData){

                         s += order.getSession_id()+"    "+order.getClose_date()+"
"+order.getSession_create()+"   "+order.getSession_name()+"
"+order.getTotalApplicants()+"\n";

                    }

                    //tv.setText(s);

               //tv.setText("Status1 :" + soapEnvelope.getResponse());

               } catch (Exception E) {

                    E.printStackTrace();

               }

               Thread.sleep(5000);

               Log.i("ARRAY", "" + m_orders.size());

          } catch (Exception e) {

               Log.e("BACKGROUND_PROC", e.getMessage());




          }
```

```
                runOnUiThread(returnRes);
        }
   private static final String SOAP_ACTION="http://tempuri.org/Isessions/close";
        private static final String METHOD_NAME = "close";
        private static final String NAMESPACE ="http://tempuri.org/";
        private static final String URL="http://69.64.43.144:8090/sessions.svc?wsdl";
        @Override
        public void onClick(View v) {
                // TODO Auto-generated method stub
                setResult(RESULT_OK);
                finish();
        }
}
```

8.2.3.

```
package com.Goggle;
import java.io.BufferedInputStream;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.URL;

import java.net.URLConnection;

import android.app.Activity;

import android.app.Dialog;

import android.app.ProgressDialog;

import android.content.Intent;

import android.os.AsyncTask;

import android.os.Bundle;

import android.util.Log;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

public class DownloadFile extends Activity {

            static int count;

    public static final int DIALOG_DOWNLOAD_PROGRESS = 0;

    private Button startBtn;
    private ProgressDialog mProgressDialog;

    Intent myIntent = null;

    /** Called when the activity is first created. */

    @Override



    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.main);

        startDownload();

    }

    private void startDownload() {

        String url = "http://www.goggle.com/uploads/superadmin.pdf";

        new DownloadFileAsync().execute(url);

    }

    @Override

    protected Dialog onCreateDialog(int id) {

        switch (id) {

            case DIALOG_DOWNLOAD_PROGRESS:

                mProgressDialog = new ProgressDialog(this);
        mProgressDialog.setMessage("Downloading file..");

        mProgressDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);

        mProgressDialog.setCancelable(false);

        mProgressDialog.show();

        return mProgressDialog;

    default:
```

CHAPTER 9

CONCLUSION

CONCLUSION

Working on the project was good experience. I understand the importance of Planning and designing as a part of software development. But it’s very difficult to complete the program for single person.

Developing the project has helped us some experience on real-time development Procedures.

BIBLEOGRAPHY

- ✓ *Used Internet(www.Google.com)*
- ✓ *Learn Android,http://code.google.com/android/*
- ✓ *XML,http://www.w3schools.com/xml/default.asp*
- ✓ *XML,http://www.tizag.com/xmlTutorial/*
- ✓ *Android forum, http://www.anddev.org/*