

Nhi Ngo

ANDROID SOFTWARE DEVELOPMENT

Case: “Logo Quiz World” Mobile Application

Thesis

CENTRIA UNIVERSITY OF APPLIED SCIENCES

Information Technology

September 2019

ABSTRACT

Centria University of Applied Sciences	Date September 2019	Author Nhi Ngo
Degree program Information Technology		
Name of thesis ANDROID SOFTWARE DEVELOPMENT Case: "Logo Quiz World" Mobile Application		
Instructor Kauko Kolehmanen		Pages 62
Supervisor Kauko Kolehmanen		
<p>"There is an application for that", this phrase is quite famous nowadays. It is true that there is a mobile application for almost anything today. People are moving from computers to mobile devices and as a result, mobile applications are becoming important in daily lives.</p> <p>The purpose of the thesis was to create a mobile application from the beginning to the final product, including user interface, Android coding and testing. This project aimed to create a small puzzle game to test the ability of users to recognize famous logos and brand names. "Logo Quiz World" is possible to be published on Android mobile devices via Play Store.</p> <p>The application has been successfully implemented and meets all the requirements set for it. There are still some parts, such as counting points or the highest score that can be developed in a way that users will be able to see their points. In addition, language settings can be developed that user will be able to switch to another language.</p>		
Key words Android Software Development, Coding, Java, User Interface Design		

CONCEPT DEFINITIONS

AVD

Android Virtual Device

SDK

Software Development Kit

XML

Extensible Markup Language

UI

User Interface

UX

User Experience

GUI

Graphical User Interface

ABSTRACT

CONCEPT DEFINITIONS

CONTENTS

1 INTRODUCTION.....	1
2 PART 1: USER INTERFACE DEVELOPMENT	2
2.1 Tools	3
2.1.1 Adobe XD.....	3
2.1.2 Adobe Photoshop	4
2.2 Design Process	5
2.2.1 User Flow	5
2.2.2 Wireframes	7
2.2.3 Logo & Colour Palettes	13
2.2.4 Mock-ups.....	16
2.2.5 Prototypes	17
3 PART 2: ANDROID DEVELOPMENT	18
3.1 Introduction	18
3.2 Tools	19
3.2.1 Android Studio	19
3.2.2 Android Virtual Devices (AVDs)	20
3.3 Android Application Description.....	21
3.3.1 Quality Function Deployment	21
3.3.2 Use Case Diagram	22
3.3.3 Manifest Files.....	23
3.4 GUI (Graphical User Interface).....	24
3.5 Activites.....	25
3.5.1 Main Activity	25
3.5.2 Activity A	26
3.5.3 How To Play.....	27
3.5.4 Game Mode.....	28
3.6 Array Adapters.....	29
3.7 Drawable Files	30
4 TESTING.....	31
4.1 Genymotion.....	31
4.2 Result Of Implementation	32
5 CONCLUSIONS	33
6 REFERENCES.....	34

1 INTRODUCTION

The Android OS is currently the most popular operating system in the world (Wallace 2017). The Android OS runs on almost everything, from smart phones to table tablets and TV. Therefore, it is not a bad idea to use this operating system to make a mobile phone application. In the market of various mobile apps in the app store, creating an app not only with an eye-catching user interface but also with a smooth user experience is not an easy job. This project includes how the mobile application “Logo Quiz World” is created from sketches to final product.

Mobile application is a fundamental component of any business nowadays and their development will improve business visibility and bring more users. It also works as another source of advertising and marketing for the business. One of the best features about mobile applications is that it is simple to use. Users only need to download, launch and use them whenever they want. Users can start using the application right after downloading.

Developing a mobile application is not an easy job. Usually, there will be many departments in a company that will work together to create an application, for example: UI/UX designers and mobile developers. UI/UX designers are a term for designers to create the user interface of a website or mobile application and ensure that users are happy to use it. Mobile developers are the ones who will create the application and make sure the application runs correctly by writing programming language (Java). Creating an application is not as simple as a desktop website and cannot be developed without experienced developers. The thesis is about how to design user interface and create a mobile application through Android Software Development. The thesis is mainly divided into two part: design UI and coding Android.

2 PART 1: USER INTERFACE DEVELOPMENT

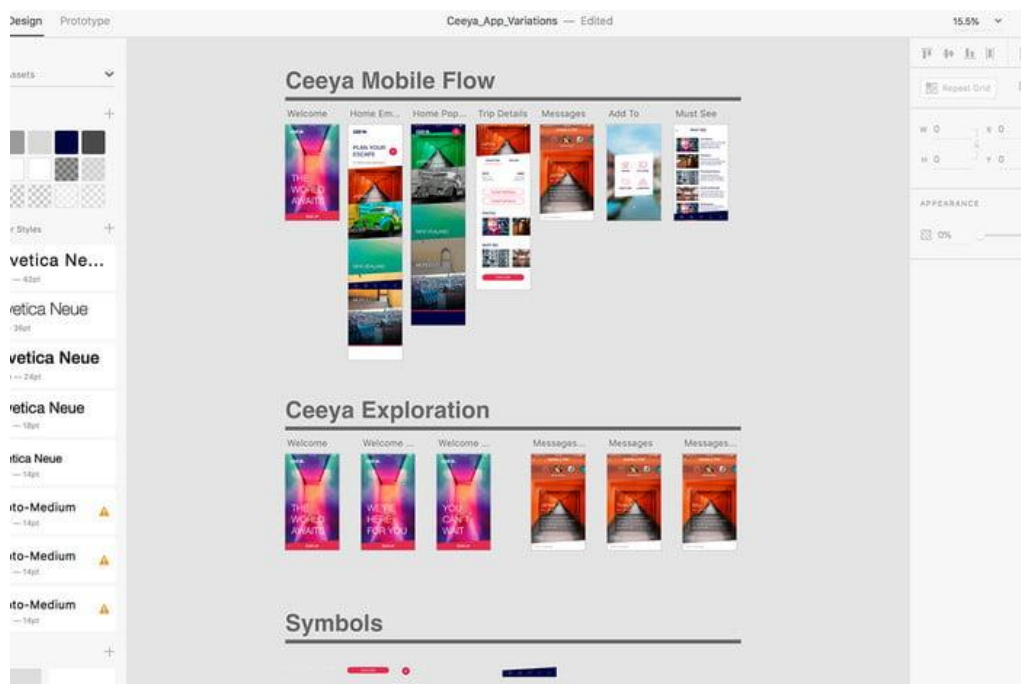
When it comes to customer's satisfaction, not only for using the website but also using mobile applications, most companies pay attention to UI/UX design. UX Design refers to the term User Experience Design, while UI Design stands for User Interface Design (Lamprecht 2019). Both factors are important for a product and work closely together. In which UX Design is more analytical and technical, UI Design is closer to what we call graphic design, although responsibilities are somewhat more complicated.

UI Development is the process of focusing on the look and the interactivity for all types of digital products like websites, applications, and other interactive devices. The usual user interface includes screens, pages, buttons, logos and other visual elements that help users interact with digital products. If you want your website, desktop or mobile app to be able to compete in today's market, it's got to be simple, intuitive and fun (Bieller 2019).

2.1 Tools

2.1.1 Adobe XD

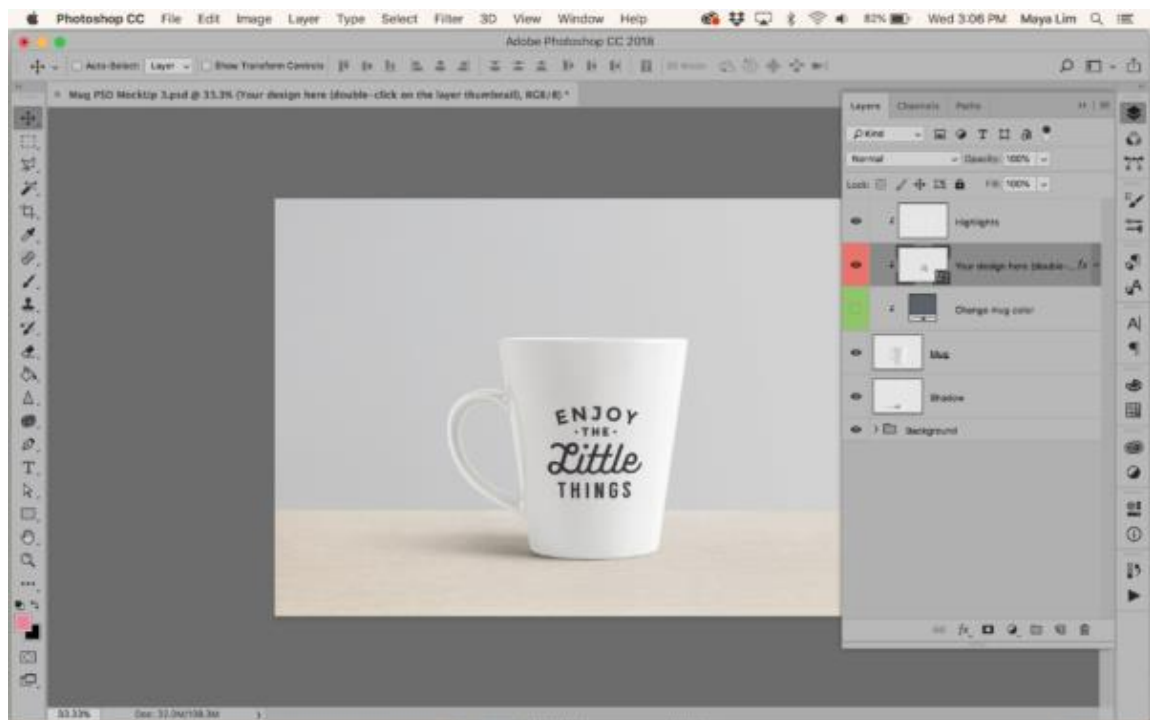
When it comes to UI development, there are a lot of applications or software available nowadays free of charge or with prices. Adobe XD, Adobe Photoshop... provides one of the best environments for digital projects in Adobe Creative Cloud design tools. Adobe XD is one of the most used free tools for UI/UX design and prototyping in the mobile design world (Rodriguez 2018). It is a vector-based tool for designing, prototyping user experience templates for websites and mobile applications, creating simple interactive clicks through prototypes. The software is available for macOS, Windows, iOS and Android for free.



PICTURE 1. Adobe XD For Making User Interface (Grigonis 2018)

2.1.2 Adobe Photoshop

Photoshop mockup is your design model image in a realistic context. Whether you are presenting your work to the customer or repetitively, mockup is a useful tool to show how the design can appear (or be used) in the real world. (Lim 2018). Presenting a mockup design for your customers is a great way to introduce them to your mobile applications or website.

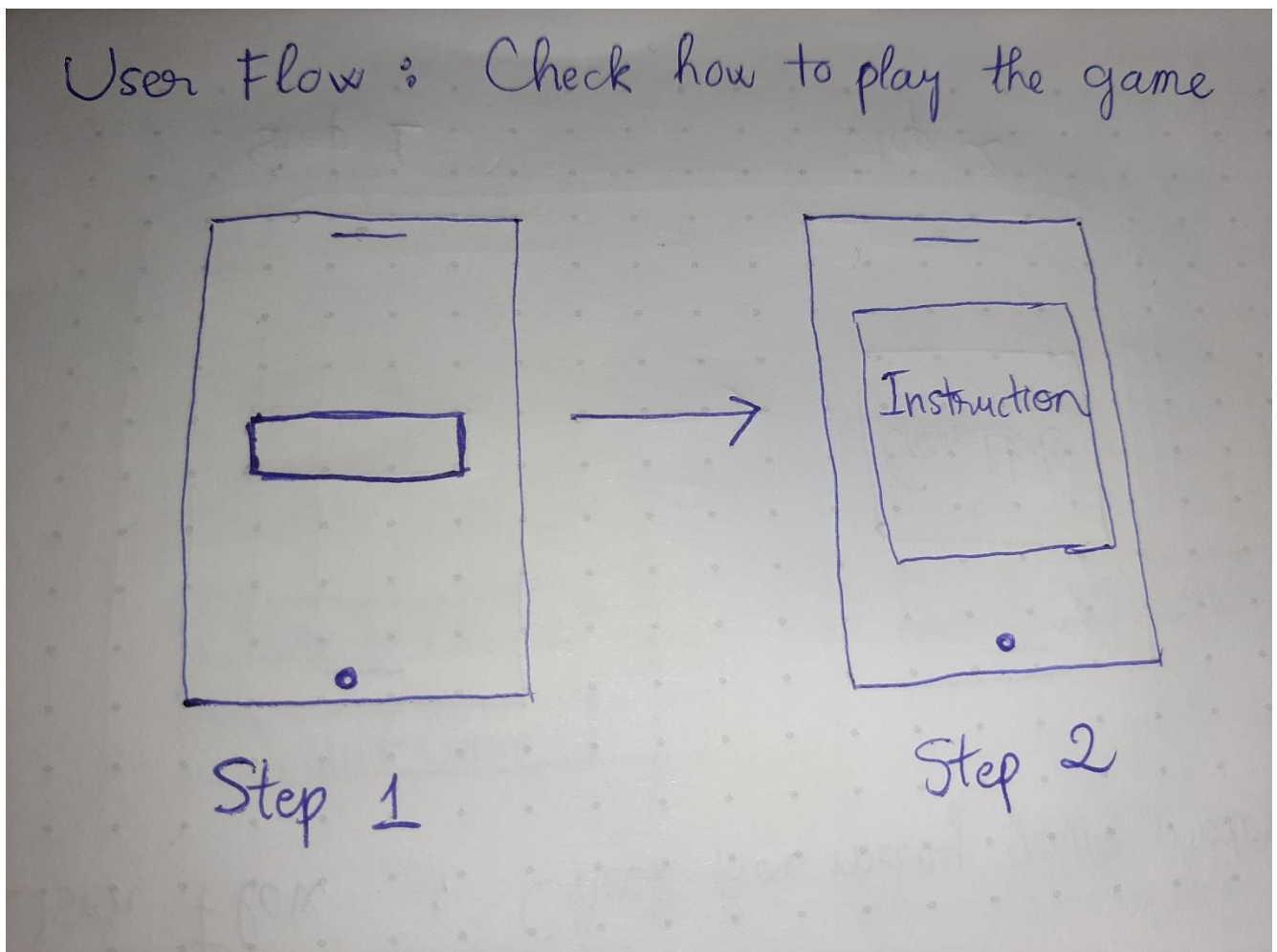


PICTURE 2. Adobe Photoshop For Making Mockups (Lim 2018)

2.2 Design Process

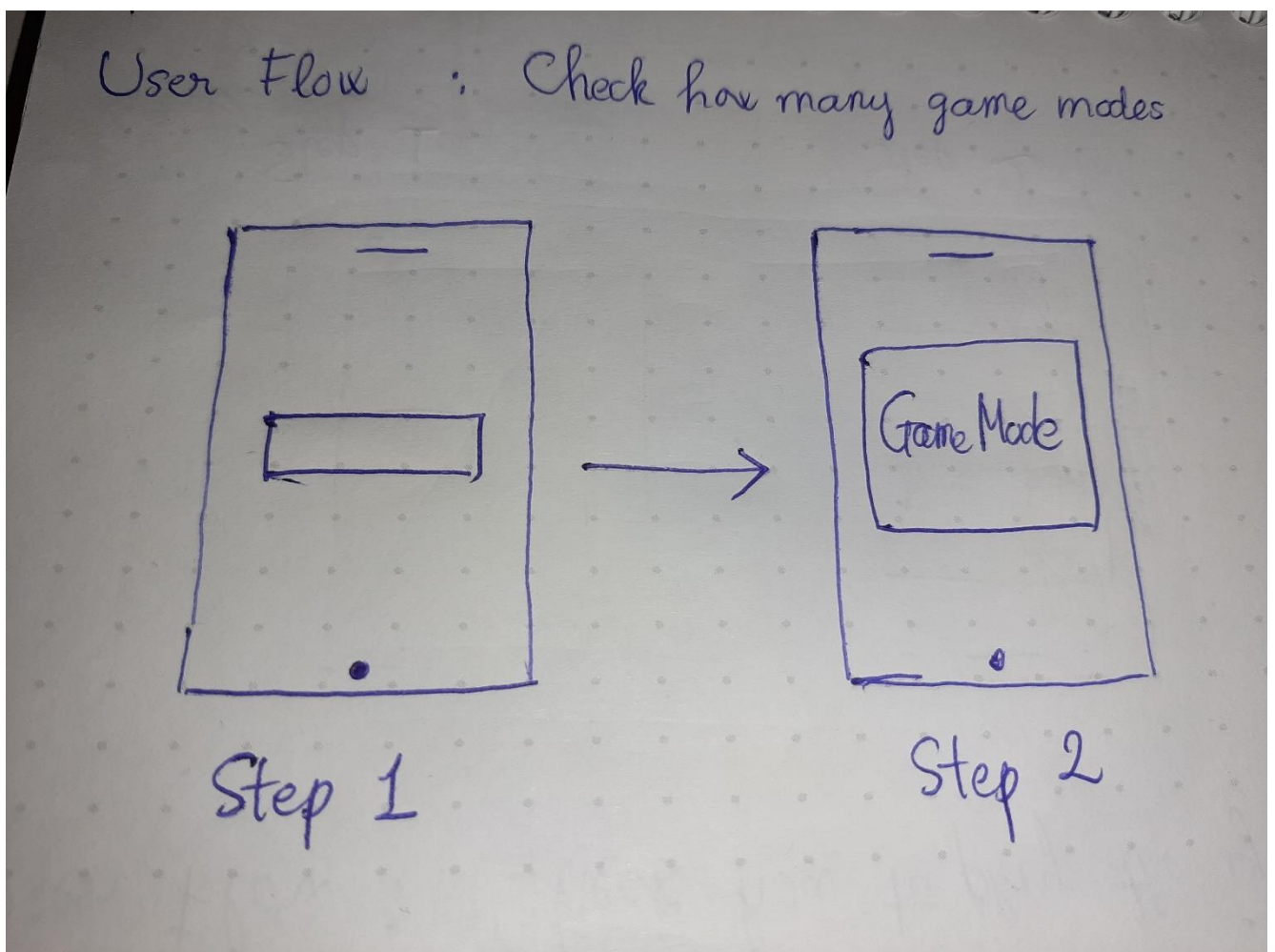
2.2.1 User Flow

Design is also about understanding your product inside out, its features and functionality, and designing while keeping the end-user in mind. (Harshita 2018). There are several steps to follow in the design process: create user-flow, wireframes, select color palettes, create mock-ups, create an animated app prototype. The first step in the design process is to find out the features you want in your application by using user flow diagrams. User flow should always tell a story.



PICTURE 3. User flow: Check how to play the game.

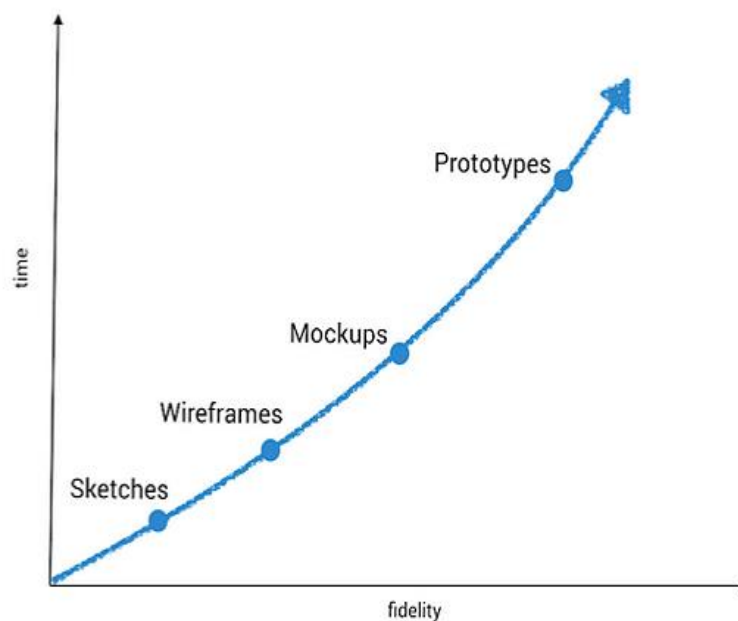
User flow is a useful element to have a large picture of how the application works. A user flow usually includes a name, steps, users, and a description of what happens at each step (Alexander 2018). There are a few basic principles to be followed to create a great user flow. First of all, user flows should describe their purpose: user goals achieved by completing the steps. Naming it is vital factor and is often overlooked. Secondly, user flows should go in one direction. User flows can have various paths but only to display different states, not different targets. To be able to break points from the confused sitemap and the prototypes can be clicked on freely, user flow should be heading on one direction and restricting decision points. Lastly, user flow performs an accomplish task. The range of user flow should be just a single task or user's target to be able to keep the design of apps or website arranged. In case that the user flow is fragment only, it will lose the ability to tell user's stories. However, if they continue for too long then they will lose their meaning.



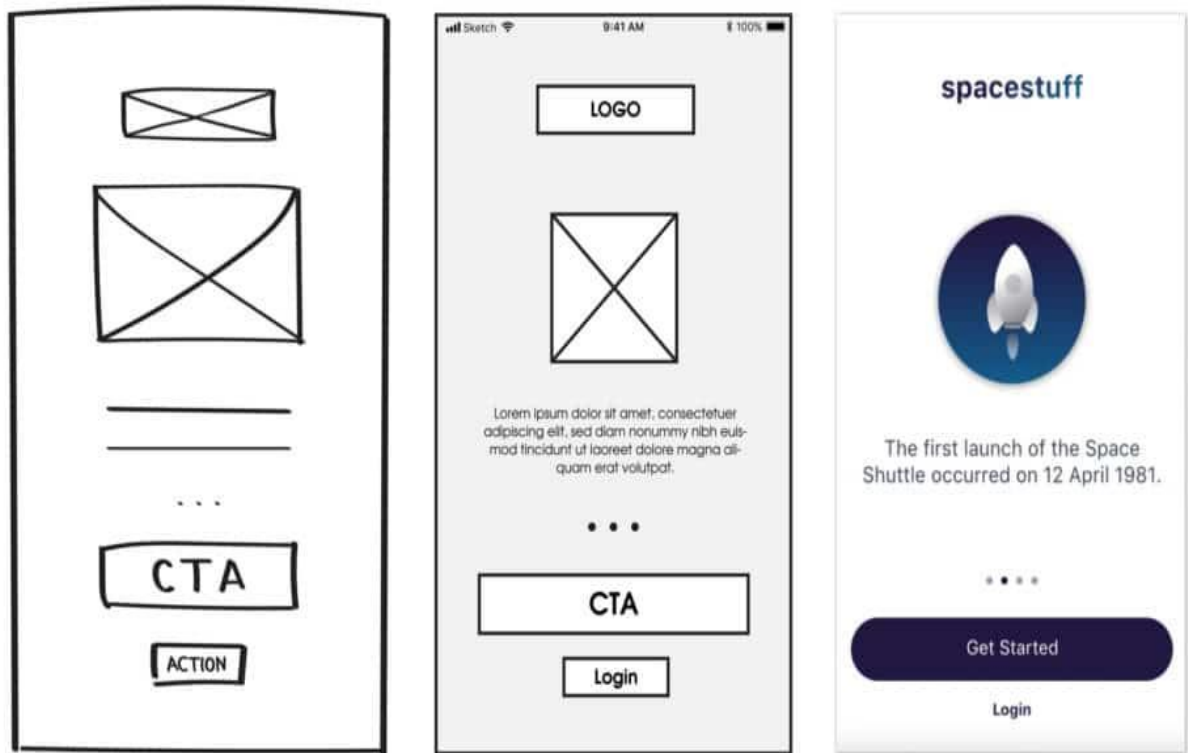
PICTURE 4. User flow: Check how many game modes.

2.2.2 Wireframes

The second step in design flow should be wireframing. Wireframe, a low-fidelity way to present a product, can efficiently outline structures and layouts (Vincent 2017). Wireframe is a basic visual representation of the layout for the website or application interface. It will deserve the time you spend, saving a lot of time and effort for you and the project members. Wireframes bring many purposes besides being a rough sketch of your application. It can help developers see the location of the elements on the page. They provide their team members to quickly plan and validate the value of page or screen design before any major effort is made to design and build the same page. Since the goal of wireframing is focusing on the structure, not to visualizing details of the design so keep it simple. Wireframes are usually limited to monochromatic palettes, which rely on different gray tones to convey the difference, in which only boxes and lines represent copies, images and page's elements... Images should not be used in wireframing to avoid distraction. It is usually represented by a rectangular box with an "X" inside. Typography is also not part of the wireframing process. You should only use one common font and still be able to change the font size to illustrate important titles and text.



PICTURE 5. Wireframe in the different stages of design flow (Vincent 2017).



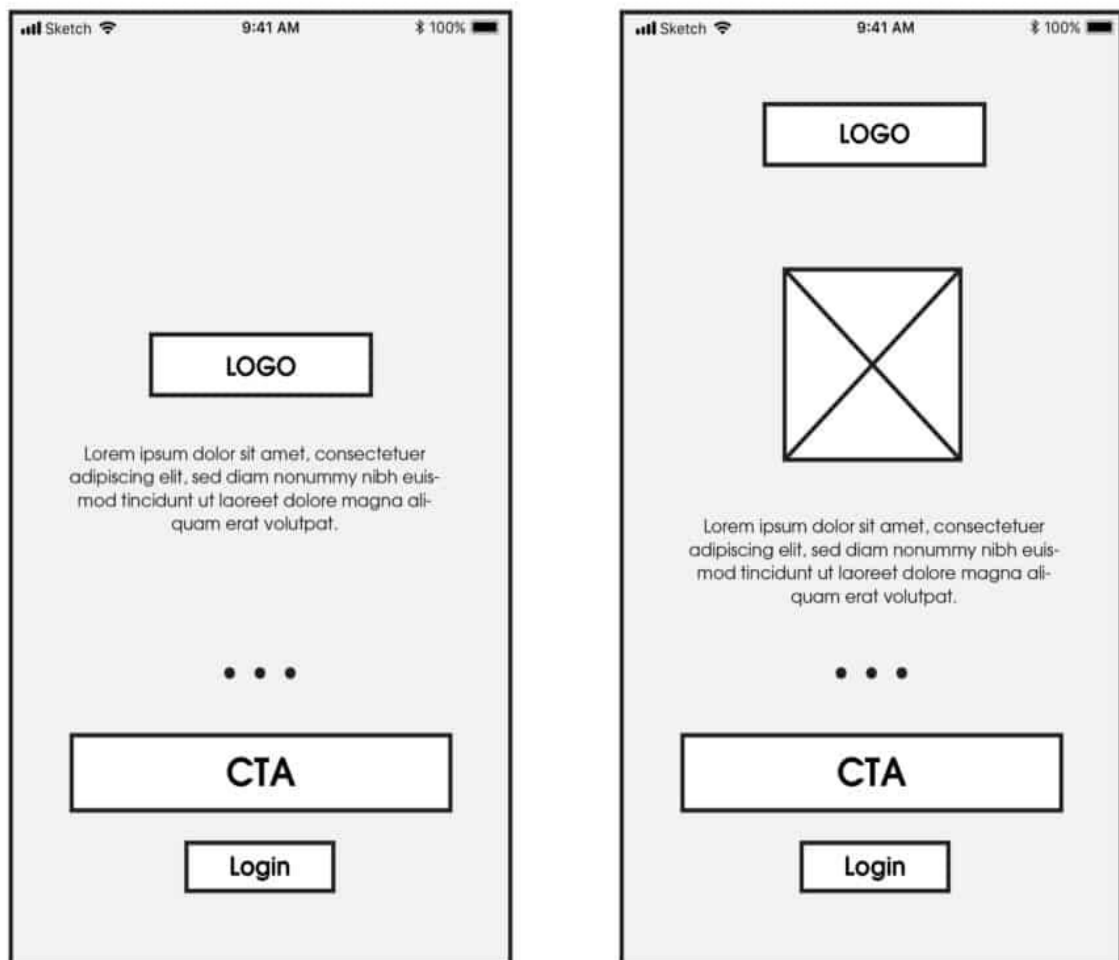
PICTURE 6. Low – Medium – High fidelity wireframe. (Lazarova 2018)

There are 3 different types of wireframes and each can be useful, depending on the progress of the project. These terms: low, medium or high fidelity is used to classify the level of wireframe production. Low fidelity grids are just a quick sketch that can make ideas more tangible (Lazarova 2018). A low-fidelity wire frame is usually a quick, simple conceptual sketch made enough to help customers know the skeleton of the design. With their simplicity, they are quick and easy to change before the next design and development efforts take place. Below are the low-fidelity wireframes and high-fidelity wireframes of the application "Logo Quiz World".



PICTURE 8. Low & High-Fidelity Wireframe Of “How To Play” Section

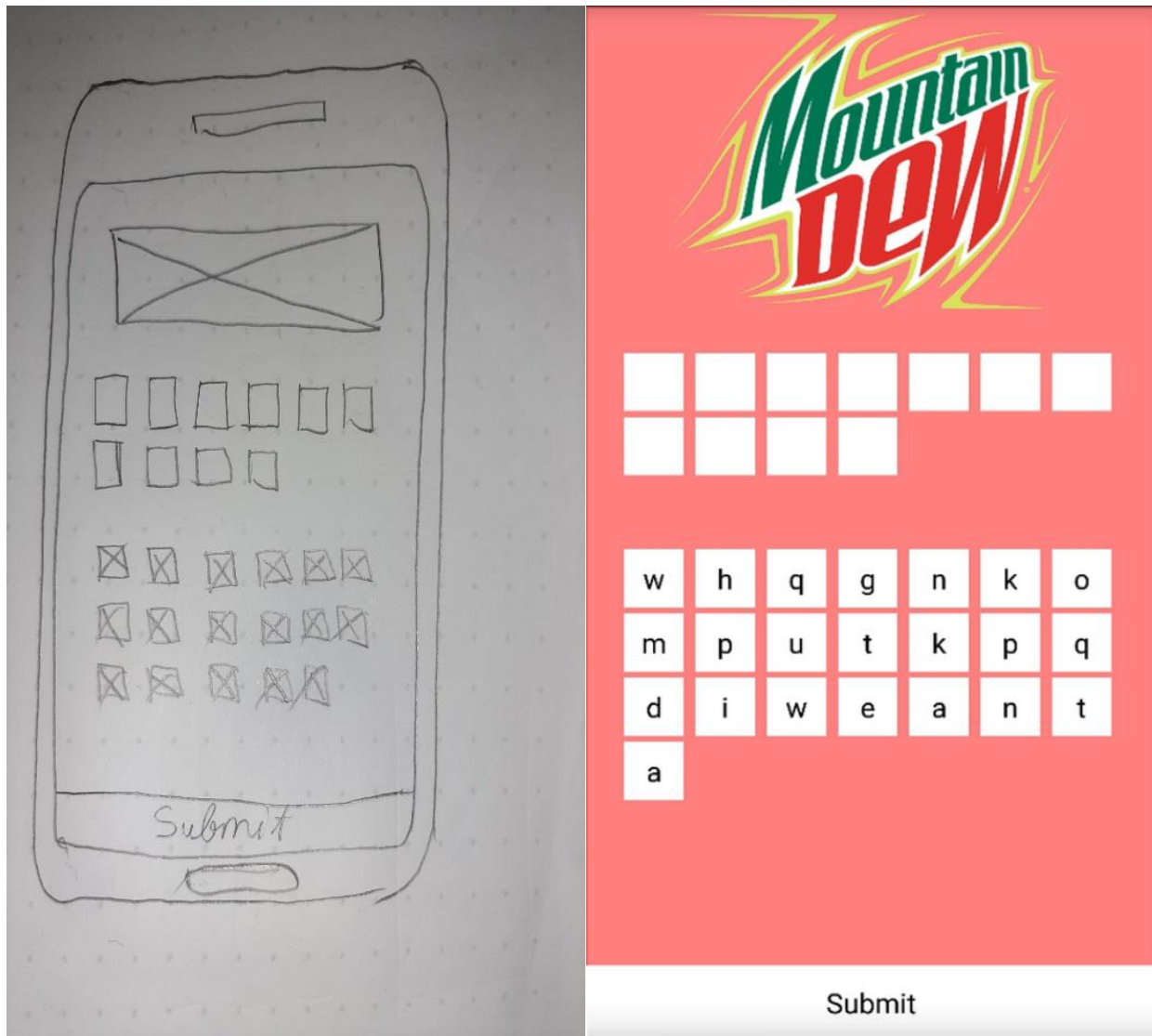
Next one in the type of frames is medium-fidelity wireframes. Medium fidelity wireframes can help to communicate to teams how aesthetic features can support essential functionalities (Lazarova 2018). Medium fidelity wireframes are created in black and white color or grayscale palettes. Medium fidelity wire frames can also be made by hand or using digital tools. Digital tools help make them more accurate and realistic user interface components.



PICTURE 9. Examples Of Medium Fidelity Wireframes (Lazarova 2018)

When you need to gather feedback, find and troubleshoot UI issues, low fidelity wireframes are no longer relevant. This is a good time to bring high-fidelity wireframes on the table. High-fidelity grids require more time and effort to make, but they work because they show how a product will look like when completing the project. The core difference from the other types of wireframes is that high fidelity wireframes are built in with color and present screens that are closer to how they would appear in final version of the software (Lazarova 2018). High-fidelity grids can only be created with a digital program, for example: Adobe XD, Sketch, Adobe Photoshop, Figma... for a more suitable design and closer look to the final product.

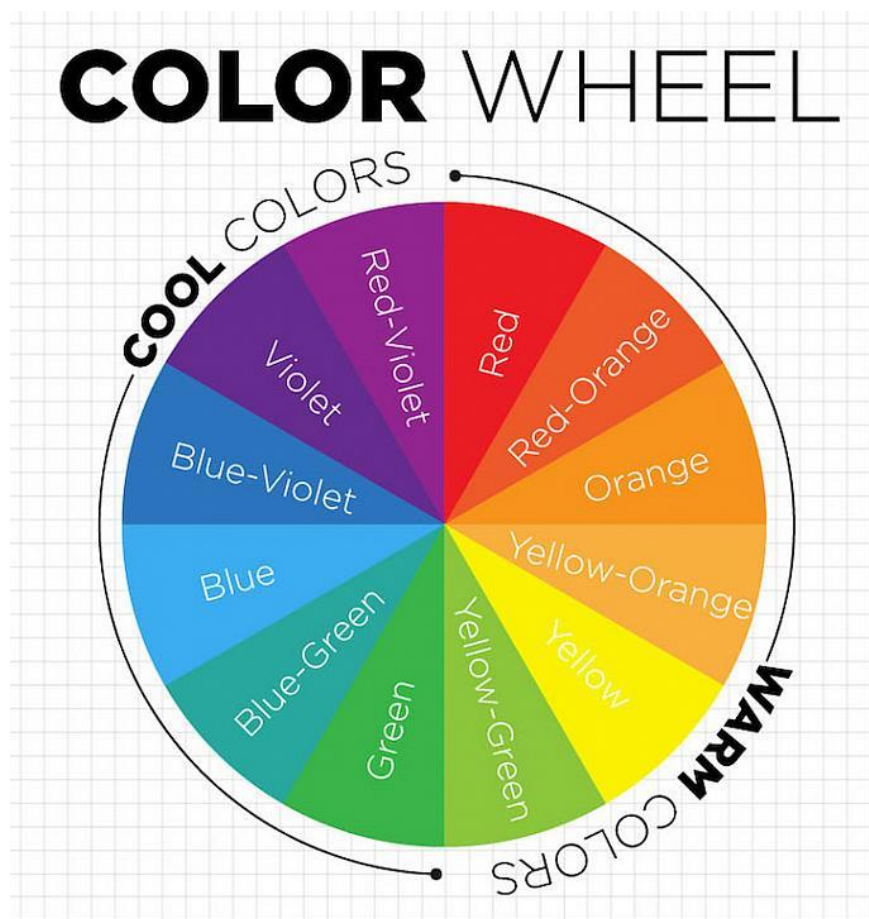
There are some main benefits of using high fidelity wireframes in the UX design process. There are impressive interaction and practical aesthetics. High- fidelity grids help test specific user interface components and interactions within one monitor or between different screens. Moreover, high fidelity wireframes often look like the final product. This will bring more detailed comments during the test. No matter what type of wireframe you are using in the design process, these are important steps when starting the design process. Without using them, there are many risks of duplications or confusions. Therefore, the user testing is no longer accurate and the person must bear the bad website or bad mobile application is your users.



PICTURE 10. Low & High-Fidelity Wireframe Of “Answering Quiz” Section

2.2.3 Logo & Colour Palettes

Color is one of the important elements of design. Understanding color theory can make a good impact on a design or brand. Have you ever wondered how designers or artists find the perfect color combination? The answer is color theory. The color wheel was invented in 1666 by Isaac Newton, who mapped the color spectrum onto a circle (Smyrnova 2019). A color wheel (color circle) is the basic of color theory, a visual representation of colors arranged according to their chromatic relationship. The two types of color wheels are RYB wheels (red, yellow, green) and RGB wheels (red, green, blue). There are many color combinations for designers to find color harmony by using color wheel. For example: Complementary, Monochromatic, Analogous, Triadic...



PICTURE 11. Color Wheel (DecoArt 2017)

Color has the power of persuasion. Learning the basics of the color wheel and how colors relate to one another will help you create color schemes that both make sense and are pleasing to the eye (DecoArt 2017). It can attract your attention, change your mood and plays a major part in how we see or define things. The meaning of colors varies depending on the culture and the impact of the brand on your target audience. Colors can increase brand identity by up to 80%. This means that colors affect the way customers perceive the brand's personality. Human brains love recognizable brands, which makes colors a major factor in creating a brand identity. For example: White represents purity and cleanliness. Green is synonymous with calm, freshness and health. Red evokes a passionate and visceral response.

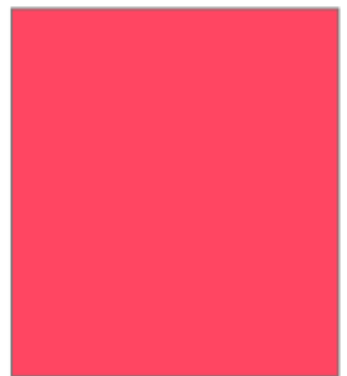
In natural senses, color is the most influential, followed by Shape, Symbol and Words. Brands that are best known for their recognition by color. However, brand recognition will make a good company succeed faster and make bad companies fail faster. By choosing colors and combining colors for your brand, it's the first step to starting your business as well as your brand.



#030303



#B2453A



#FF4662

PICTURE 12. Color Palettes Of “Logo Quiz World”

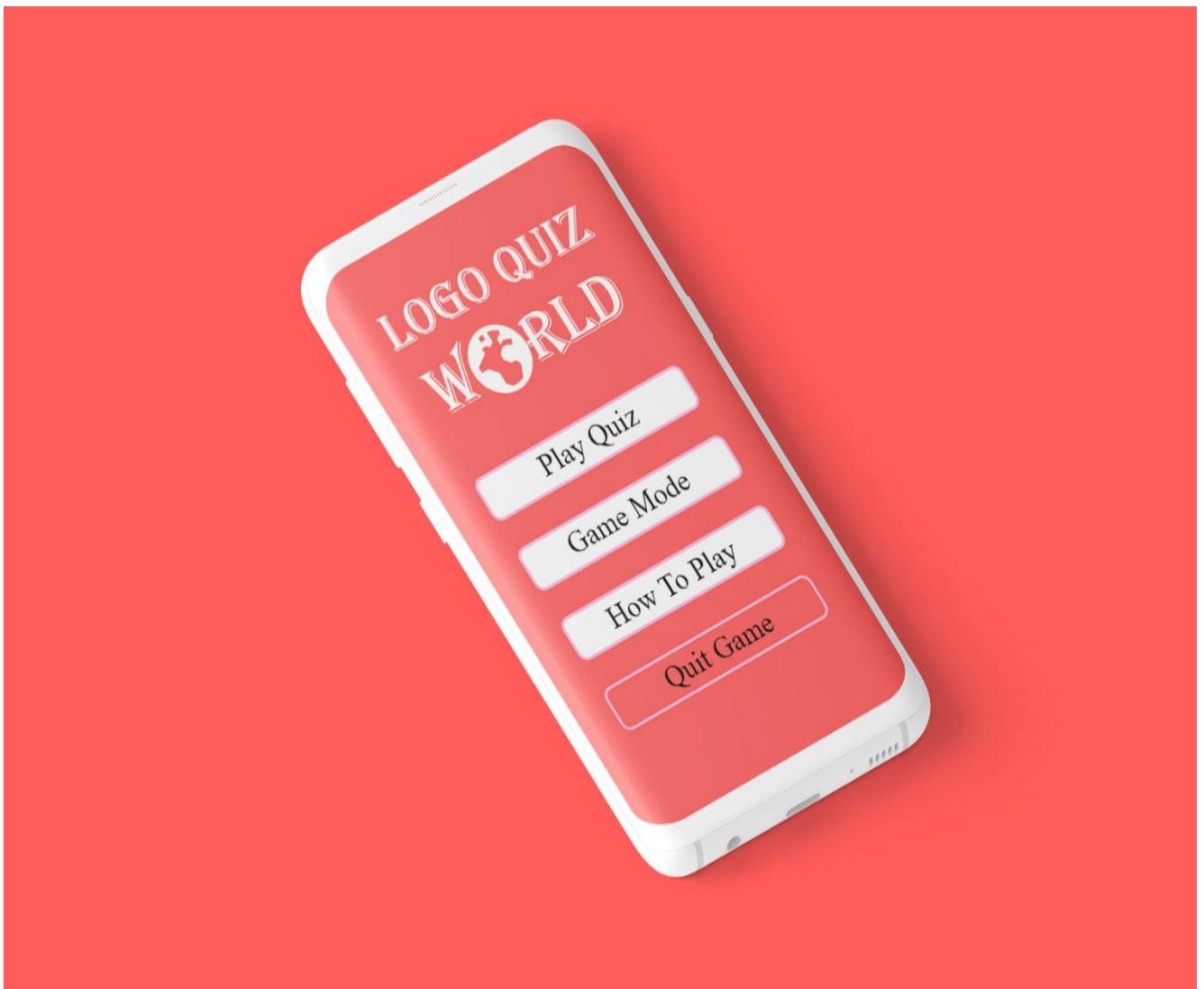
One of the important things when you start a business is to create a memorable logo. A logo can be represented as the face of a company. If people connect with your branding, the likelihood is they'll be more open to whatever it is offering them (Carson, Airey 2019). There are many reasons why logos are so important. It can attract customers' attention and give a first impression of the brand. A good logo can be brand identity and help stand out from other brands. Below is the application logo with vivid and friendly colors. The "O" in "World" will be replaced by the symbol of the global earth.



PICTURE 13. Logo Of The Application “LOGO QUIZ WORLD”

2.2.4 Mock-ups

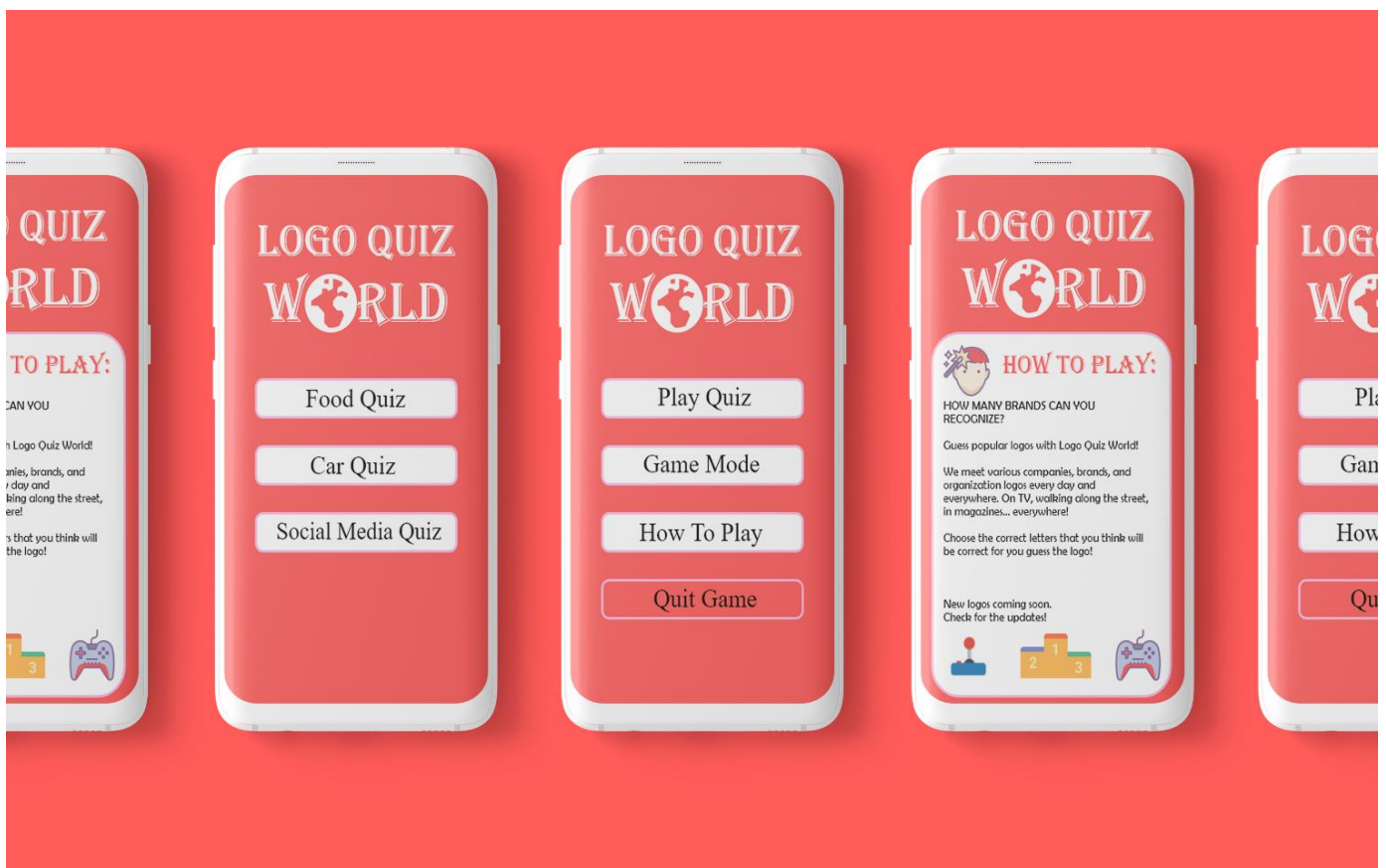
The next formative stages of the UX design process is making the mock-ups. A mockup is usually a medium to high fidelity representation of the product's appearance and demonstrates the basics of its functionality (Cao 2015). By looking at a mockup, you will get the closer look to what the final product will be. Although mockup seems to resemble the final design, it lacks the functionality of a prototype. However, mockup is a channel between wireframes and prototype.



PICTURE 14. Mock-Up Of The Main Page Of “LOGO QUIZ WORLD”

2.2.5 Prototypes

The final stage of the process is prototyping. Prototypes enable designers, clients, and test users to validate the usability and overall value of solution design — but in terms of design, they are still different from the final product itself, and this distinction is key (Lazarova 2018). Prototypes are clickable images and an effective way for designers to approve their work and test their design before changing from user experience design to user interface. In addition, these clickable images can allow customers or test groups to be able to test and provide feedback until the final product is completed.



PICTURE 15. The Complete User Interface Of The Application

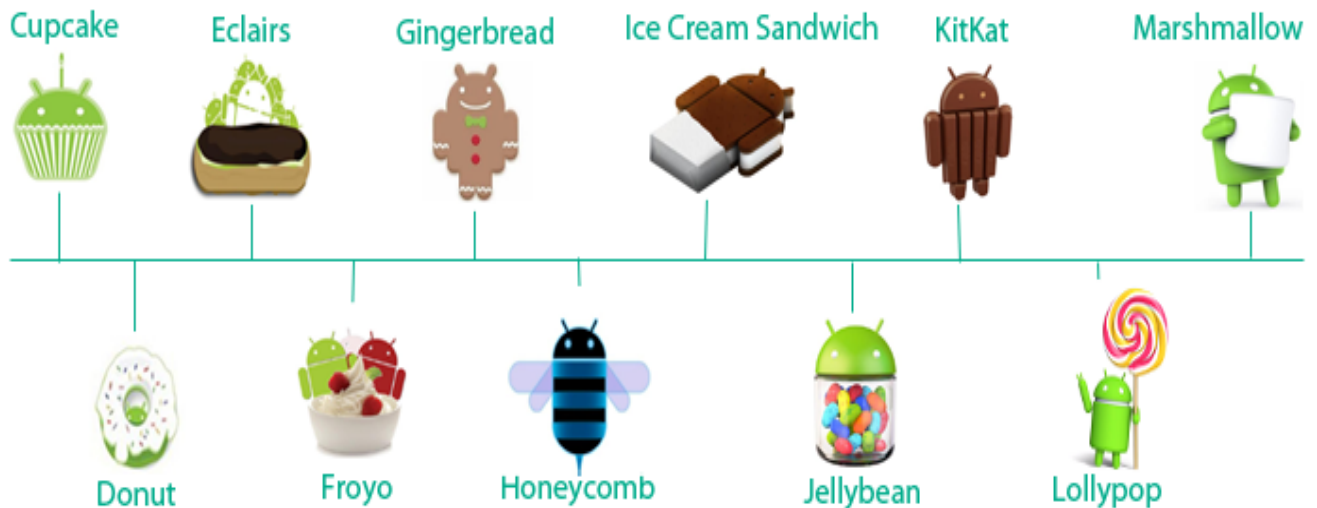
3 PART 2: ANDROID DEVELOPMENT

Google Android is a stack of software for mobile devices including the operating system, middleware, and key applications. Android is an open source and it uses Linux to provide core system services. The Android OS is currently the most popular OS in the world today, running on everything from watches to HD smartphones to touchscreen tablets to eBook readers to interactive television sets. (Wallace 2013).

In general, Android applications are written in Java (Kotlin). Android SDK tools compile the files into an Android package, an archive file with an .apk suffix. An application consists of many core components. Each Android application lives in its own security sandbox. By default, every application runs in its own Linux process. Each process has its own virtual machine, so the code of one application runs separately from other applications (but possibly IPC).

3.1 Introduction

It is worth noting that Google names all Android versions alphabetically, meaning that after Google 9.0 Pie (released in 2018), the next Android name starts with the letter Q. On the other hand, except Android 1.0 and 1.1, all other Android versions are named after sweets or desserts. Many people prefer using these codenames instead of using the version numbers (Dabade 2015). People are always looking forward to the new version of Android every time. Each new version has more updates and more useful features than the old version.



History of Android Operating System

PICTURE 16. History Of Android Versions (Dabade 2015)

3.2 Tools

3.2.1 Android Studio

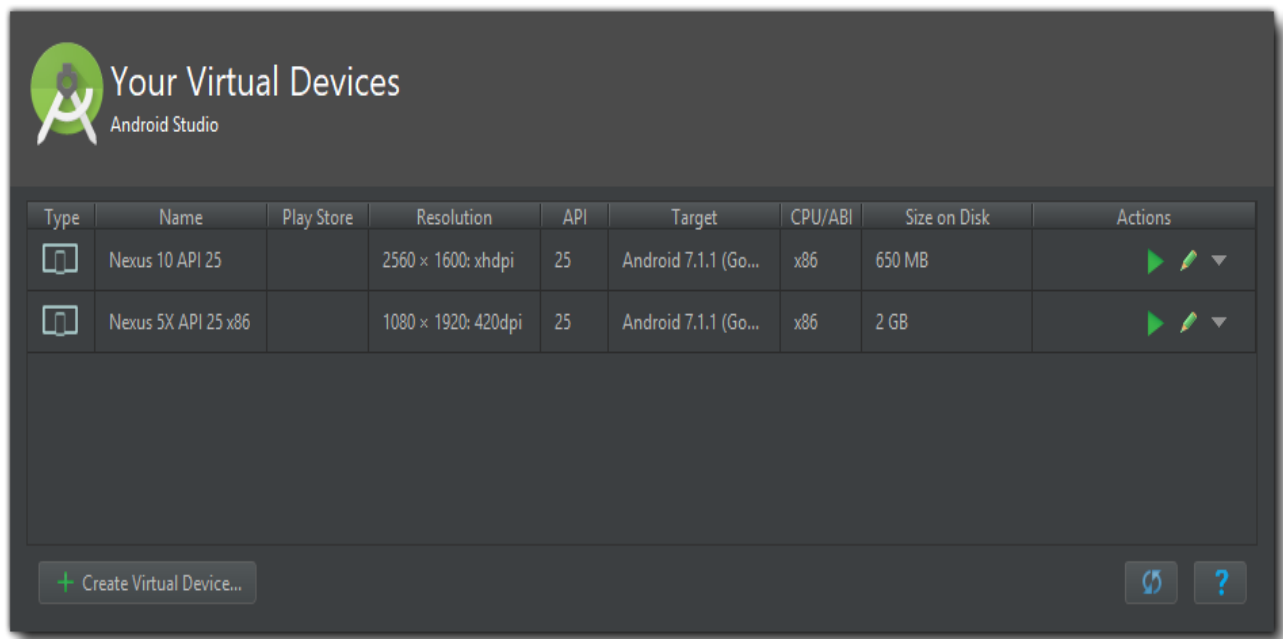
The first thing to do is to choose the right working environment to develop the programs. Android Software Development Kit is the answer for developers. There are many options for using, for example, Android Studio - the official integrated development environment (IDE) developed by Google, Unity and GameMaker: Studio. Android Studio is the official integrated development environment (IDE) for building recognized Android applications for the Google Play Store. Android Studio and Software Development Kit, which can be downloaded straight from Google, may need some power to run evenly. The tool needs to allow you to view and edit many different files, some of which operate in completely different ways (Sinicki 2018). Along with code editing features, Android Studio includes a simulation software to run applications as they work so that developers can do everything they need without leaving the environment. Android Studio is one of forceful tools for developing entirely useful and testing Android applications.



PICTURE 17. Android Studio Working Environment (Sinicki 2018).

3.2.2 Android Virtual Devices (AVDs)

Another convenient feature of Android Studio is the Android Virtual Device (AVD). An AVD is an virtual machine running Android on an emulated set of device specifications, including memory, screen size, CPU, etc... (Alexander 2019). It will help developers test applications on a virtual device. There are many options that allow users to customize every form of device, such as Android version (regularly updated), memory, brand, screen size... Configure the virtual device you create, to model the typical features of the device in Android Emulator. In each configuration, you can specify the Android platform to run, hardware options and the emulation interface to use. Each AVD acts as a standalone device with separate storage for user data and SD cards.



PICTURE 18. Android Virtual Devices (AVDs) (Alexander 2019).

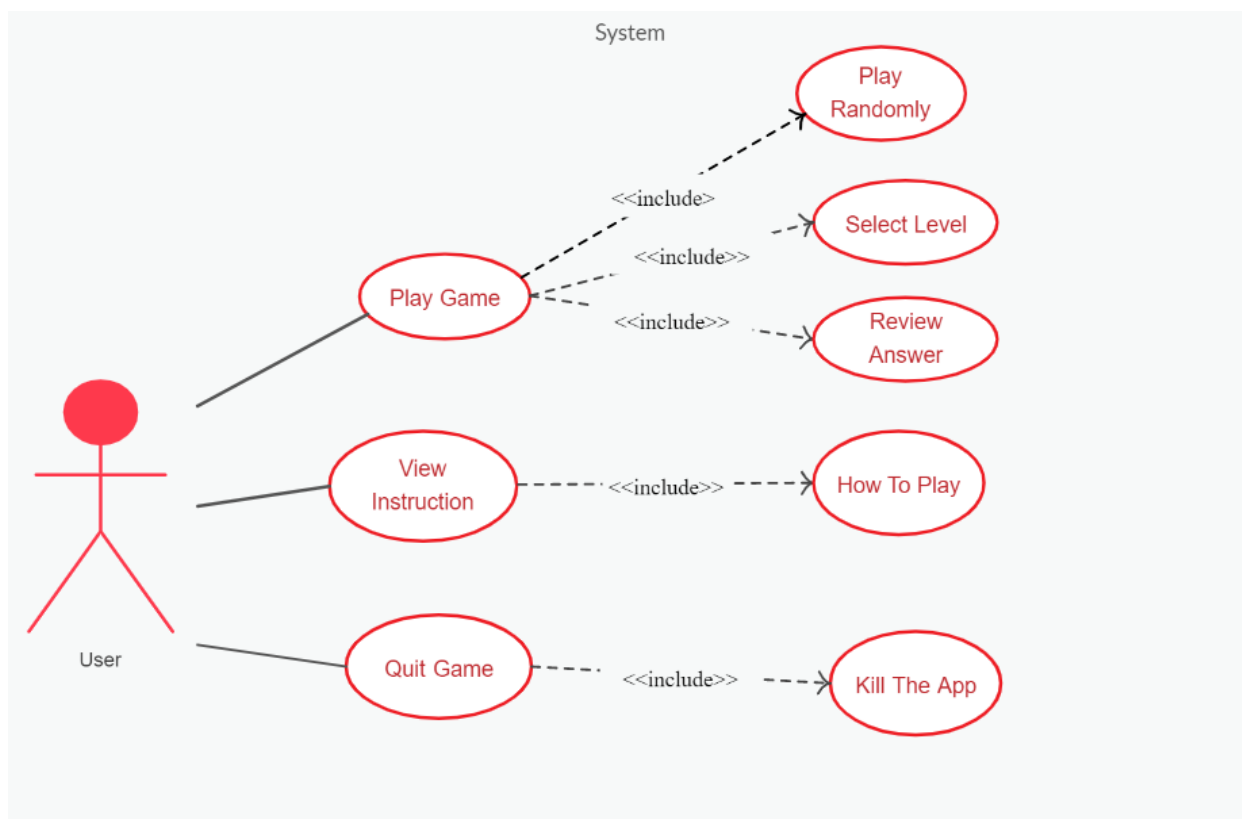
3.3 Android Application Description

3.3.1 Quality Function Deployment

There are several requirements for this application when it is fully developed. The expected requirements must be: The UI of the application will be colorful and friendly; Users will be able to play the game easily and do not need a lot of instructions; Users must be able to view instructions screen; Choose different game modes; Play the quiz in random or selected mode; See the answer of the quiz; Automatically move to the next quiz after answering the previous question correctly; There is no problem opening or using the application. Furthermore, there will be optional requirements that may be available in the future. For example: View scores (Users can check their current scores) and View scoreboard (User can check other player's scores).

3.3.2 Use Case Diagram

Use case modeling in the Unified Modeling Language (UML) is a popular text-based tool for systems analysis and design (Gemino, Parker 2009). The functions of the application can be seen from the diagram below. Users have access to all features on the application. This diagram shows what happened at each stage of the application and how it works. Users have three different game modes to play (Food Quiz, Car Quiz, Social Media Quiz). After selecting a mode, the quiz will only show the quiz from that topic. For example: If the user selects the topic "Food Quiz", only quizzes related to famous food or beverage chains such as Wendy, Mountain Dew will appear. On the other hand, if they don't want to choose a specific topic, they just need to click directly on the "Play quiz" button, random quizzes from all topics will pop out. After answering the question, there will be a toast message showing "right" or "wrong" to the player. If the user clicks on the "How to play" button, the instruction screen is displayed. Moreover, if the user clicks to "Quit Game" button, the application will be killed.



PICTURE 19. Use Case Diagram.

3.3.3 Manifest Files

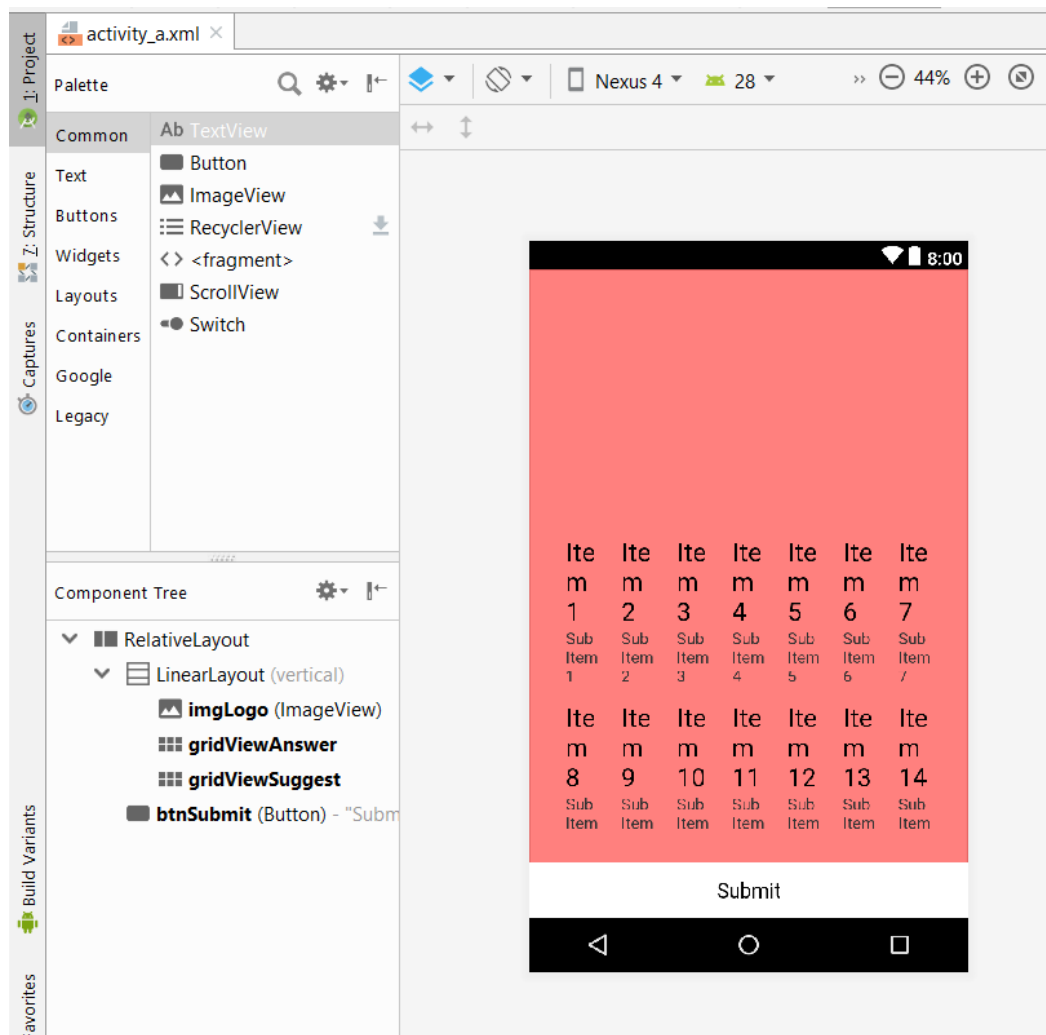
The manifest is always placed in the root directory of the application directory. It is an important file and every application cannot live without it. The AndroidManifest.xml file defines your application characteristics, theme, assets, activities, and permissions (Wallace 2017). The manifest file also contains some other important features, like Activity, Service, Receiver, Provider... Declaring an Activity (a subclass of Activity) implements an intuitive user interface of the application. Each activity must be represented by its own <Activity> element in the manifest. Any undeclared activity will not be seen by the system. Declaring a Service (a Service subclass) is one of the application components. Each Service must be represented by its own <service> element in the manifest. Any undeclared service will not be seen by the system. The BroadcastReceiver declaration (a subclass of BroadcastReceiver) is one of the components of the application. The ContentProvider (subclass ContentProvider) provides structured access to data managed by the application. Every ContentProvider that is part of the application must be represented by a <provider> element in the manifest.



PICTURE 20. "Logo Quiz World" Manifest Files.

3.4 GUI (Graphical User Interface)

The graphical user interface (GUI) is an important aspect of any application. The GUI is an interface that uses icons or other visual indicators to interact with electronic devices, instead of just text via a command line. The user selects one or a combination of these elements on the user interface by pressing keys on a keyboard, clicking with a mouse, or tapping on the screen. (Teske 2019). Each screen is designed with xml. It contains all the buttons to the important features embedded into the application. Every screen is designed with user friendly interface and colorful buttons. The Activity A below in the picture is for play randomly mode and made with linear layout (vertical). There will be the image of the logo (ImageView) above and gridViewAnswer and gridViewSuggest in the middle. Finally, there will be a submit button (btnSubmit) at the bottom of the screen.

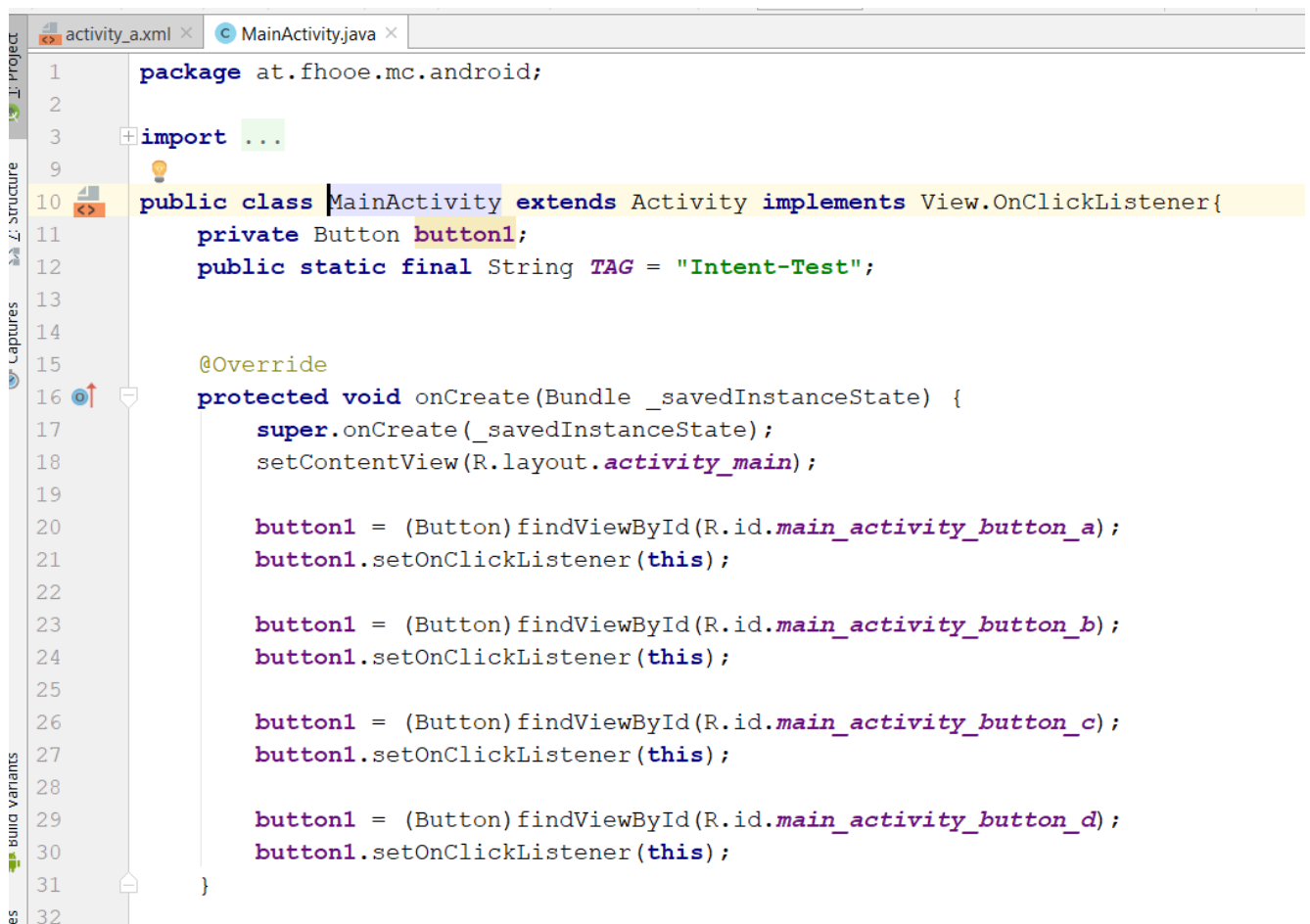


PICTURE 21. “Activity A” GUI

3.5 Activites

3.5.1 Main Activity

An Activity object in Android contains a single UI focused on a specific task or feature that your application offers to the user (Wallace 2017). Almost all activities interact with the user, so the Activity class provides a window in which one can place one's UI. An activity is created by 2 parts, an xml file that defines its interface and a Java file that contains the source code. The "Logo Quiz World" application has 6 activities. Therefore, there are 6 screens that are displayed to the user if needed. This is the first activity that users will see whenever they open the application. Each button seen in the main operation is a feature of the application. It consists of the main menu with a list of application features that users can access.



```
1 package at.fhooe.mc.android;
2
3 import ...
4
5
6
7
8
9
10 public class MainActivity extends Activity implements View.OnClickListener{
11     private Button button1;
12     public static final String TAG = "Intent-Test";
13
14
15     @Override
16     protected void onCreate(Bundle _savedInstanceState) {
17         super.onCreate(_savedInstanceState);
18         setContentView(R.layout.activity_main);
19
20         button1 = (Button) findViewById(R.id.main_activity_button_a);
21         button1.setOnClickListener(this);
22
23         button1 = (Button) findViewById(R.id.main_activity_button_b);
24         button1.setOnClickListener(this);
25
26         button1 = (Button) findViewById(R.id.main_activity_button_c);
27         button1.setOnClickListener(this);
28
29         button1 = (Button) findViewById(R.id.main_activity_button_d);
30         button1.setOnClickListener(this);
31     }
32 }
```

PICTURE 22. Main Activity Code

3.5.2 Activity A

All activity classes will have a corresponding <activity> declaration in their package Android manifest XML file (Wallace 2017). If forgetting to name an activity happens in the Android Manifest, that activity will not work even though there's no error in that activity. Activity A is used for the random quiz, which includes every quiz in the other three game modes of the app. There are lots of pictures in this activity because of the number of quizzes. Every "drawable" in image_list must be case-sensitive because there will be errors if there are wrong letter somewhere in the name of the image. For example: R.drawable.burgerking will be an error if the file in "Drawable Files" is named BurgerKing.png. It will lead to many cases, the least damage case if it comes to the "Burger King" puzzle, the photo will not show. Moreover, the worst scenario is that it will cause the application to crash when the puzzle appears.

```
package at.fhooe.mc.android;

import ...

public class ActivityA extends Activity {
    public List<String> suggestSource = new ArrayList<>();

    public GridViewAnswerAdapter answerAdapter;
    public GridViewSuggestAdapter suggestAdapter;

    public Button btnSubmit;

    public GridView gridViewAnswer, gridViewSuggest;

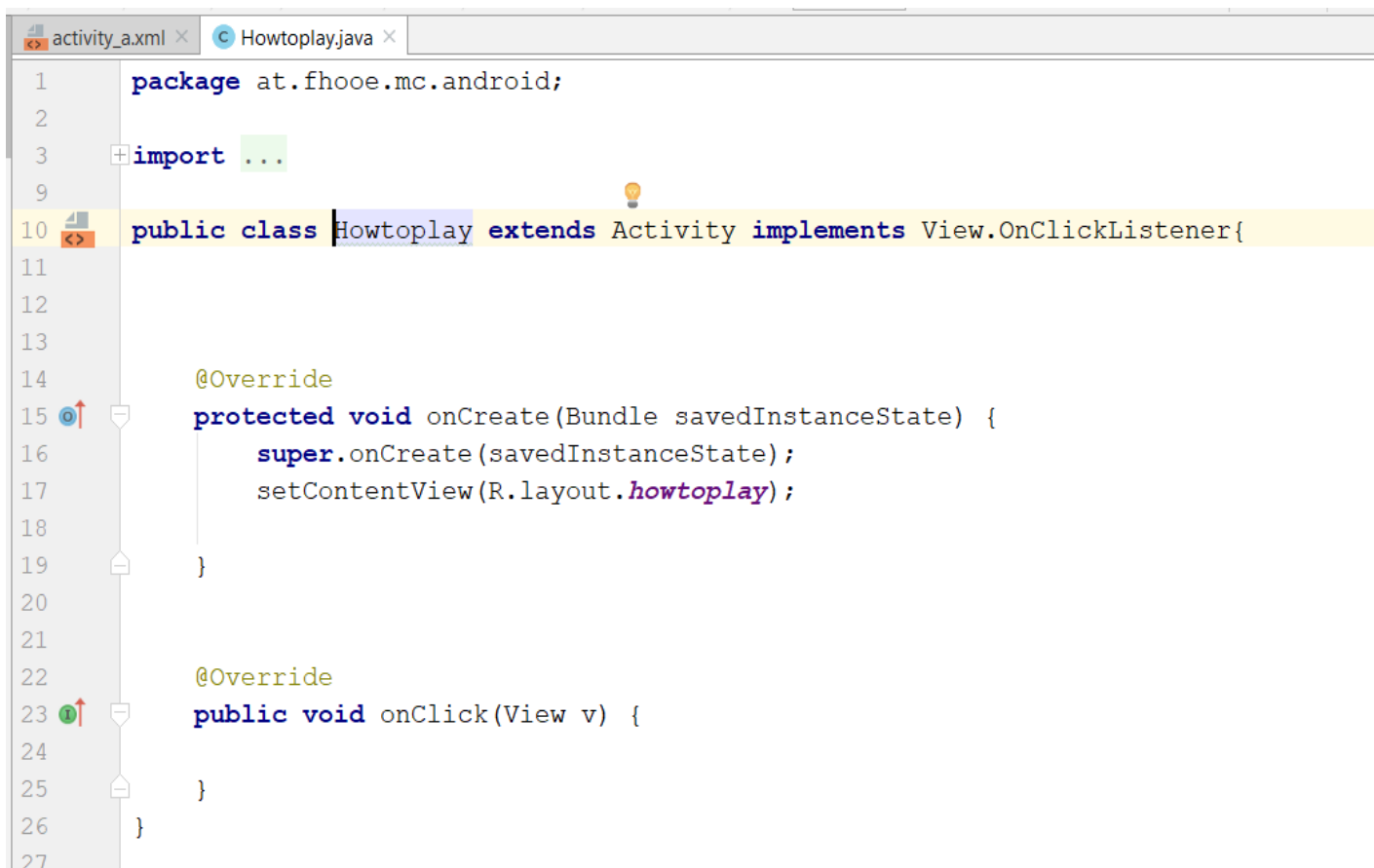
    public ImageView imgViewQuestion;

    int[] image_list={
        R.drawable.burgerking,
        R.drawable.kfc,
        R.drawable.wendys,
        R.drawable.mcdonalds,
        R.drawable.mountaindew,
        R.drawable.pizzahut,
        R.drawable.nestle,
        R.drawable.starbucks,
        R.drawable.knorr,
        R.drawable.pringles,
        R.drawable.apple,
        R.drawable.blogger,
    }
}
```

PICTURE 23. Activity A Code

3.5.3 How To Play

This activity is very simple. There is one Activity class method Android classes must implement to get your UI design screen into system memory: the `onCreate(Bundle)`, where you initialize the Activity object and use `setContentView()` to reference and load your UI design. (Wallace 2017). There is only one "intent" leads from the "How to play" button in the main screen. The purpose is a message container provided or provided by the Android system to activate components / actions or receive information.

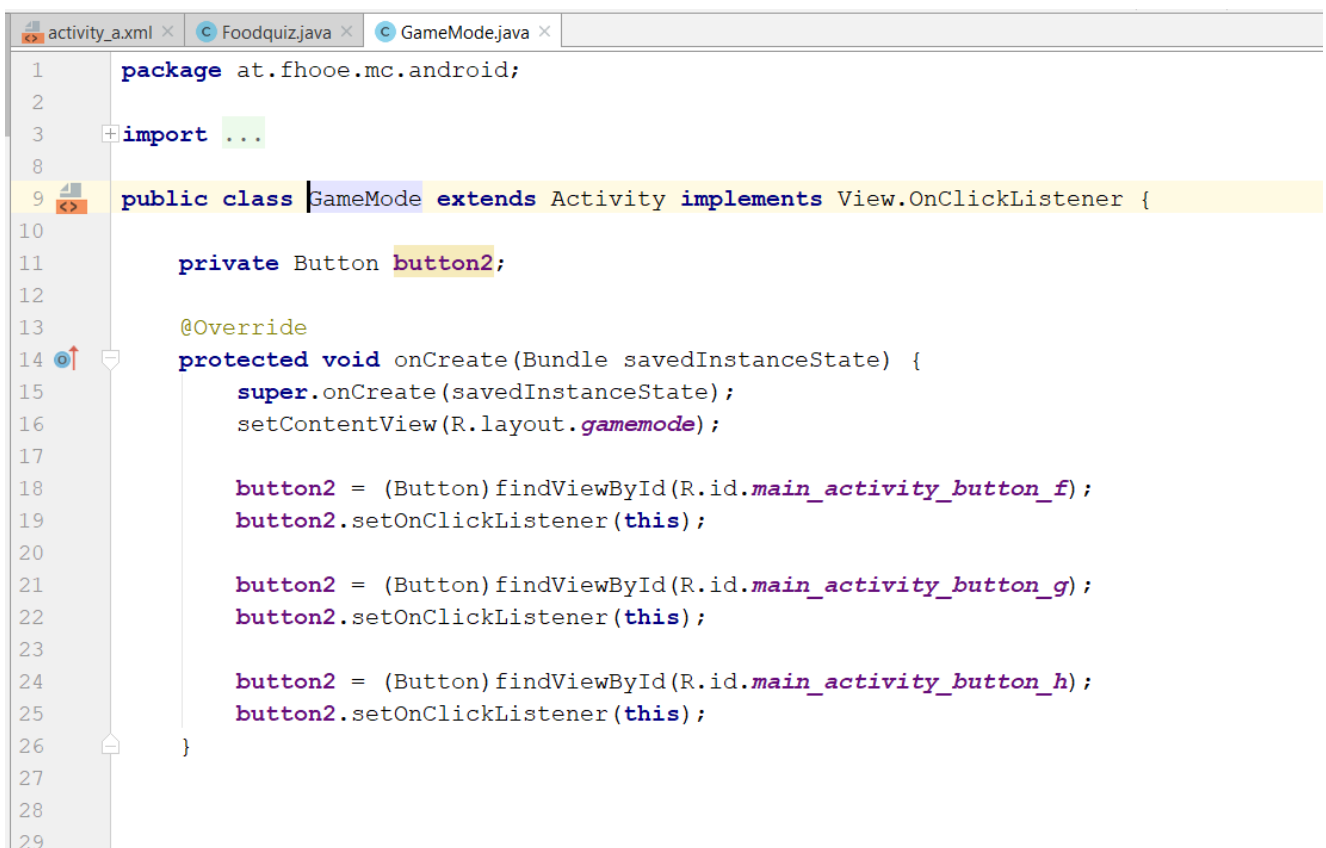


```
1 package at.fhooe.mc.android;
2
3 import ...
4
5
6
7
8
9
10 public class Howtoplay extends Activity implements View.OnClickListener{
11
12
13
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout.howtoplay);
18
19     }
20
21
22     @Override
23     public void onClick(View v) {
24
25     }
26 }
27
```

PICTURE 23. How To Play Code

3.5.4 Game Mode

In the images below, you can see the coding of the game mode part. The “.setOnClickListener()” method that sets the event listener to an implementation of View.OnClickListener that in turn contains your onClick() event processing infrastructure (Wallace 2017). There will be three buttons leads to three intents using setOnClickListener. Those three main buttons (main_activity_button_f, main_activity_button_g, main_activity_button_h) lead to three different classes. Each button has its own name and ID.

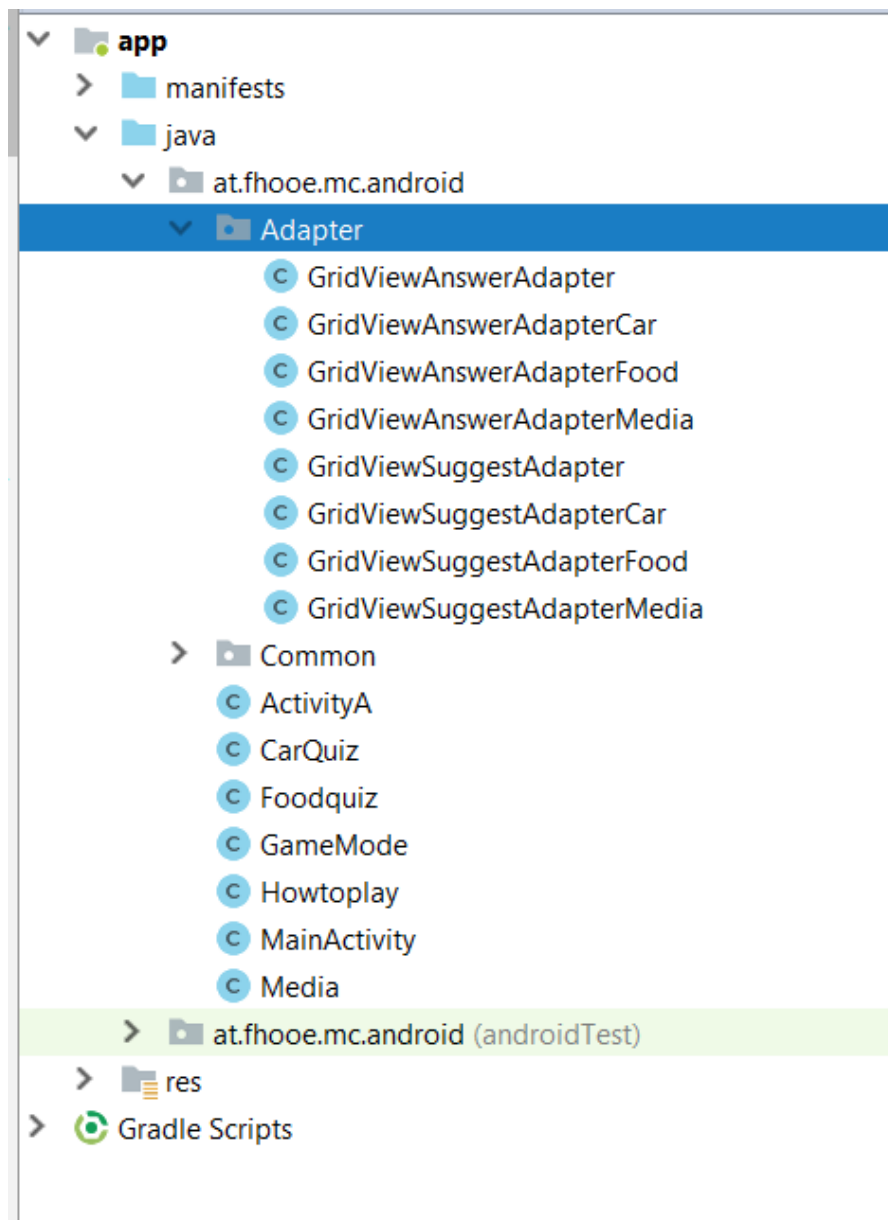


```
1 package at.fhooe.mc.android;
2
3 import ...
4
5
6
7
8
9 public class GameMode extends Activity implements View.OnClickListener {
10
11     private Button button2;
12
13     @Override
14     protected void onCreate(Bundle savedInstanceState) {
15         super.onCreate(savedInstanceState);
16         setContentView(R.layout.gamemode);
17
18         button2 = (Button) findViewById(R.id.main_activity_button_f);
19         button2.setOnClickListener(this);
20
21         button2 = (Button) findViewById(R.id.main_activity_button_g);
22         button2.setOnClickListener(this);
23
24         button2 = (Button) findViewById(R.id.main_activity_button_h);
25         button2.setOnClickListener(this);
26     }
27
28
29 }
```

PICTURE 24. Game Mode Code

3.6 Array Adapters

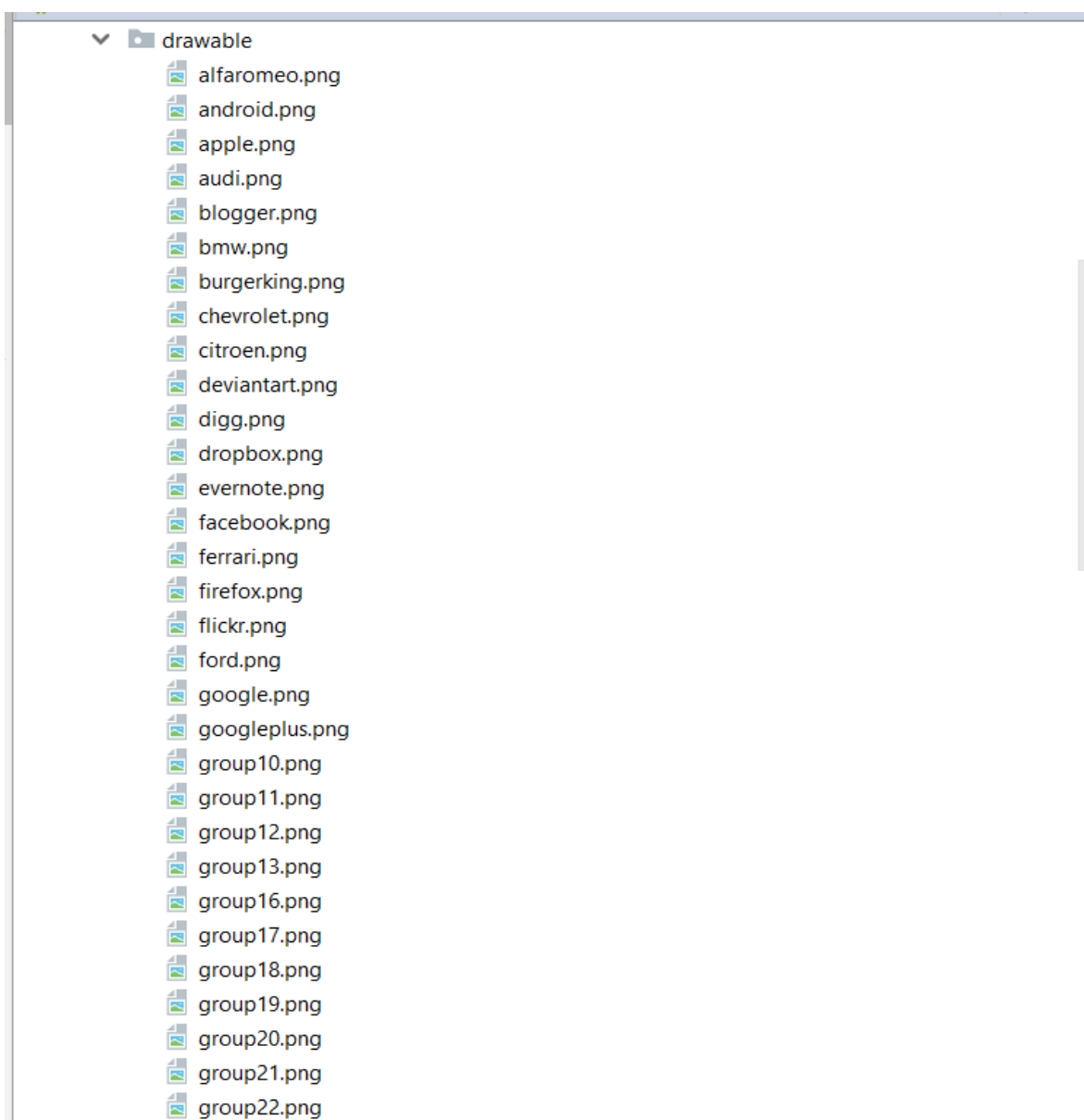
ArrayAdapter is a basic Adapter and is usually used in Android. Adapter is the link between the UI component and the data source that helps us fill in the UI component. It holds the data and sends data to the adapter view, then the view can retrieve the data from the adapter view and display the data on different views such as Listview and Gridview. This is where you process your menu item selections in your Activity's options menu, by using the `.getItemId()` method from the Adapter class (object) that is used to process lists (Wallace 2017).



PICTURE 25. Array Adapters

3.7 Drawable Files

In Android Studio, whenever you need to display static images in your application, you can use the Drawable class and its subclasses to draw shapes and images. The standard work process to define an Android multi-state ImageButton UI element is to utilize an XML Drawable definition file, which will use a <selector> parent tag and be located in the /res/drawable folder (Wallace 2017). Drawable resources are a common concept for a graphic that can be drawn on the screen and you can approach them by APIs like `getDrawable (int)` or apply to another XML resource with attributes like `android:drawable` and `android:icon`.

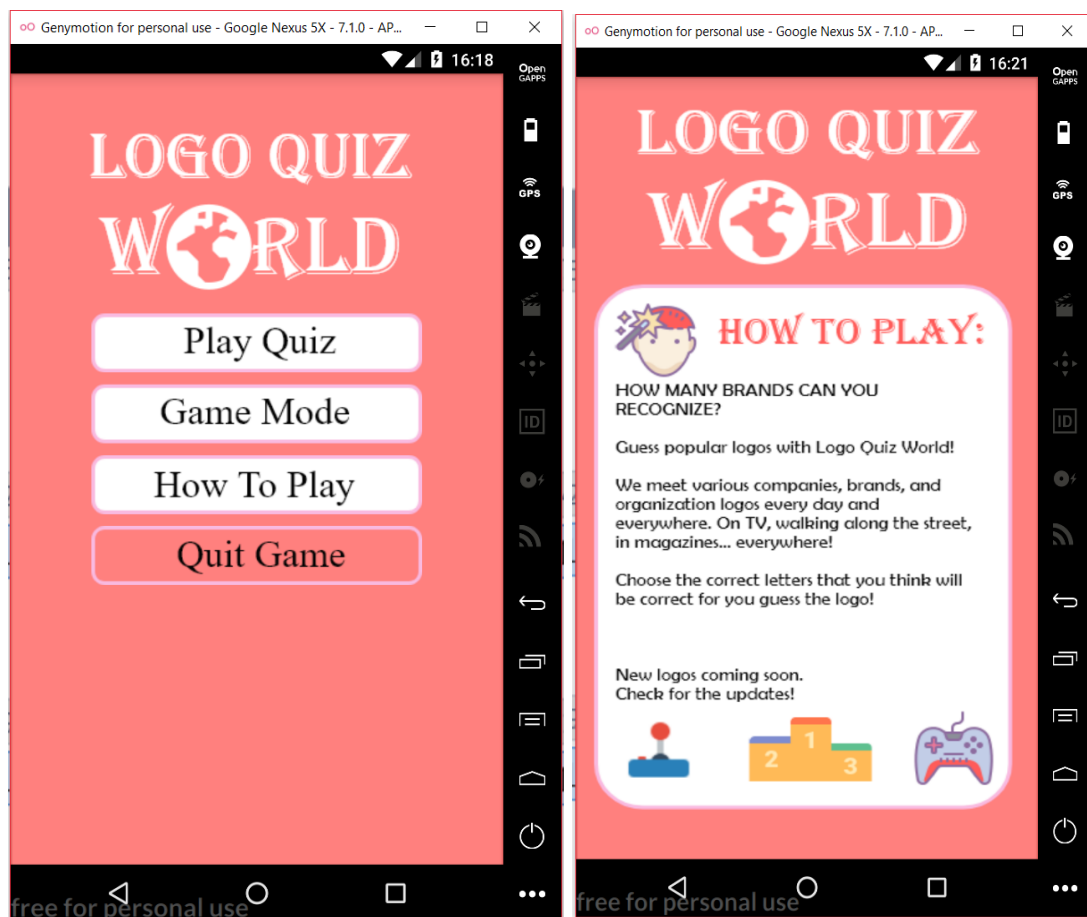


PICTURE 26. Drawable files

4 TESTING

4.1 Genymotion

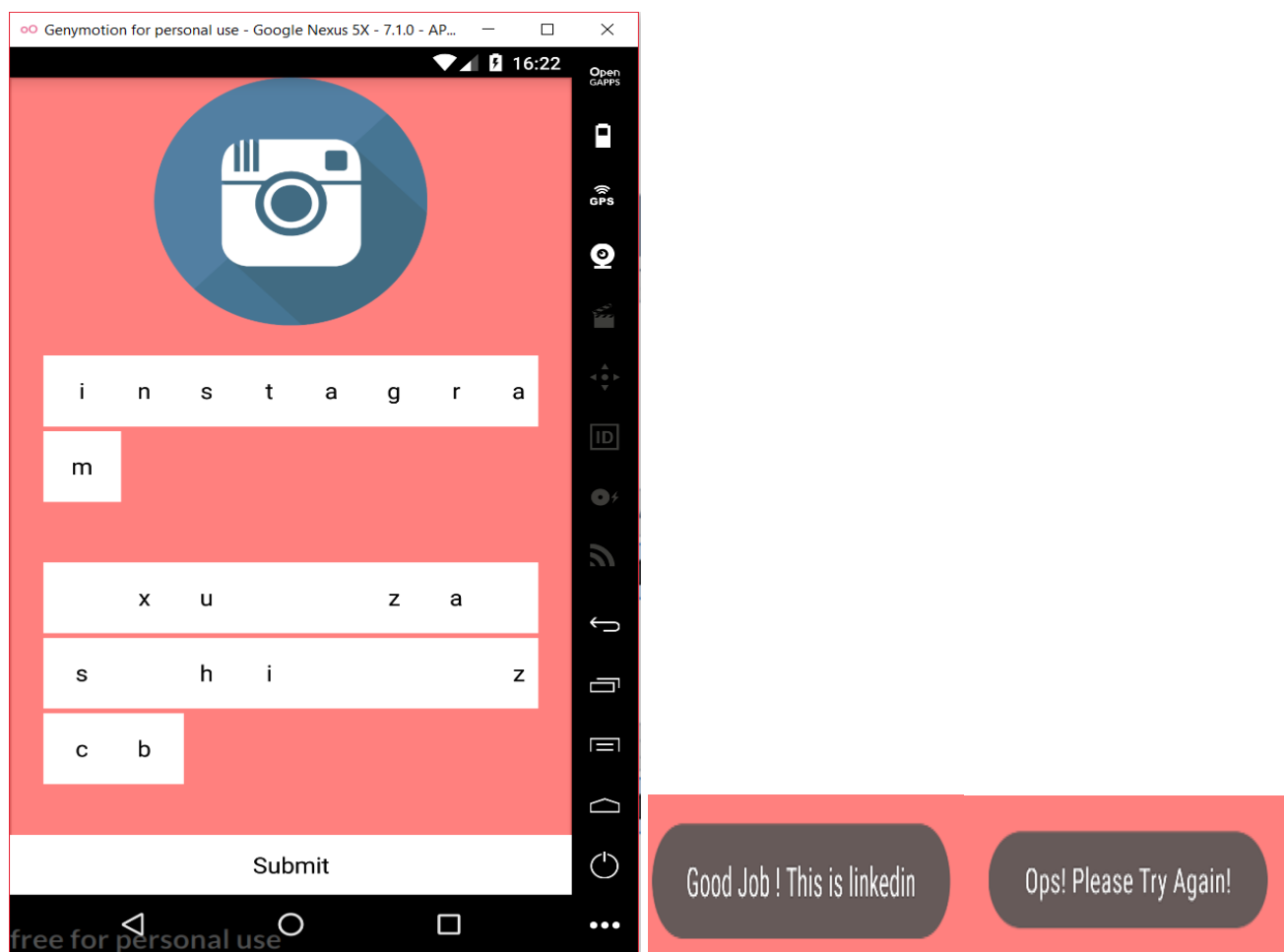
Testing is carried out along with coding using Android emulator and Android devices. After completing the coding for this application and fixing all possible errors, it is ready for final testing. Java objects make debugging easier, because you can add or remove them modularly, or isolate them during testing in order to ascertain where bugs are located within the overall code (Wallace 2017). There will be testing with virtual emulators (Genymotion) and real Android mobile devices. Genymotion is an easy-to-use Android emulator, created to help developers test their products in a safe, virtual environment. Genymotion has essential virtualization requirements. It works by establish a virtual machine through VirtualBox to give an Android emulator that supports hardware sensors like GPS, accelerometer and battery.



PICTURE 27. Testing With Genymotion Emulator (Main Page & How To Play Page)

4.2 Result Of Implementation

When developing any Android application, the project contains apk. files can be transferred in mobile devices running on Android. The files are extracted by phone when the application get installed. This project has been tested also on Samsung J3 (2016). The main page, game mode page and instruction page worked perfectly as expected as shown in the image below. New users can easily select any feature of the application without any problems. There will be a notification of a toast message pop up whenever the user clicks the submit button. It will show the user the correct answer or say: "Ops! Please try again" when the answer is wrong.



PICTURE 28. Game Quiz Page (Genymotion)

5 CONCLUSIONS

The idea of "Logo Quiz World" is to entertain and provide knowledge to users. "Logo Quiz World" is not only tested on virtual emulators but also Android mobile devices. The goal has been met, users can play the game smoothly. The application runs well and the user interface is friendly and lively as expected. The size of the text and the buttons vary slightly when installing on a mobile device. The structure of this thesis is assessed with the important results of the test that were performed on Android mobile phones. This application is designed for basic purposes although it can be developed for more complex tasks. There are a few improvements that can be made to upgrade this app, for example: view score rankings, view current scores and review questions.

6 REFERENCES

Wallace, 2017. Android Apps for Absolute Beginners. Accessed: 20th July 2019.

Rodriguez, 2018. Learning the basics of Adobe XD in one hour. Available: <https://medium.com/drill/learning-the-basics-of-adobe-xd-in-one-hour-3537f3ac02a3>. Accessed: 21th July 2019.

Grigonis, 2018. New Adobe XD Starter Plan opens user experience design to all. Available: <https://www.digitaltrends.com/computing/adobe-xd-cc-starter-plan-announced/>. Accessed: 21th July 2019.

Lim, 2018. How To Use Photoshop Mockups: Five Simple Steps. Available: <https://www.howdesign.com/creative-freelancer-blog/how-to-use-photoshop-mockups-five-simple-steps/>. Accessed: 21th July 2019.

Harshita, 2018. Designing beautiful mobile apps from scratch. Available: <https://www.freecodecamp.org/news/designing-beautiful-mobile-apps-from-scratch-1a3441ebd604/>. Accessed: 21th July 2019.

Alexander, 2018. The biggest WTF in design right now. Available: <https://uxdesign.cc/the-biggest-wtf-in-design-right-now-87139f367d66>. Accessed: 21th July 2019.

Vincent, 2017. Wireframe vs Mockup vs Prototype & Selection of Prototyping Tools. Available: <https://www.mockplus.com/blog/post/wireframe-mockup-prototype-selection-of-prototyping-tools>. Accessed: 22th July 2019.

Lazarova, 2018. Low Fidelity Wireframes vs High Fidelity Wireframes. Available: <https://mentormate.com/blog/low-fidelity-wireframes-vs-high-fidelity-wireframes/>. Accessed: 23th July 2019.

Smyrnova, 2019. Canva's Color Wheel for great color combinations. Available: <https://syndicode.com/2019/05/10/canvas-color-wheel-for-great-color-combinations/>. Accessed: 28th July 2019.

DecoArt, 2017. Color Theory Basics: The Color Wheel. Available: https://decoart.com/blog/article/318/color_theory_basics_the_color_wheel. Accessed: 28th July 2019.

Carson, Airey 2019. Logo design: everything you need to know. Available: <https://www.creativebloq.com/graphic-design/pro-guide-logo-design-21221>. Accessed: 28th July 2019.

Cao 2015. The What, Why, and How of Mockups. Available: <https://designmodo.com/mockups/>. Accessed: 28th July 2019.

Wallace, J. 2013. Learn Android App Development. Accessed: 19th August 2019.

Dabade, 2015. History of Android Operating System. Available: <http://techtrickle.com/history-of-android-operating-system/>. Accessed: 19th August 2019.

Sinicki 2018. Getting to know Android Studio and the files that make up your apps. Available: <https://www.androidauthority.com/how-to-use-android-studio-860903/>. Accessed: 19th August 2019.

Alexander, 2019. Android: Setting Up Virtual Devices. Available: <https://help.yoyogames.com/hc/en-us/articles/115000270191-Android-Setting-Up-Virtual-Devices>. Accessed: 19th August 2019.

Gemino, Parker 2009. Use Case Diagrams in Support of Use Case Modeling: Deriving Understanding from the Picture. Available: <https://www.semanticscholar.org/paper/Use-Case-Diagrams-in-Support-of-Use-Case-Modeling%3A-Gemino-Parker/beae261f69654889b929057bb0a718e888882e04>. Accessed: 19th August 2019.

Teske, 2019. What Is GUI (Graphical User Interface)?. Available: <https://www.lifewire.com/what-is-gui-graphical-user-interface-4682595>. Accessed: 19th August 2019.

APPENDIX 1/1 Mock-Up Of The Game Mode Page



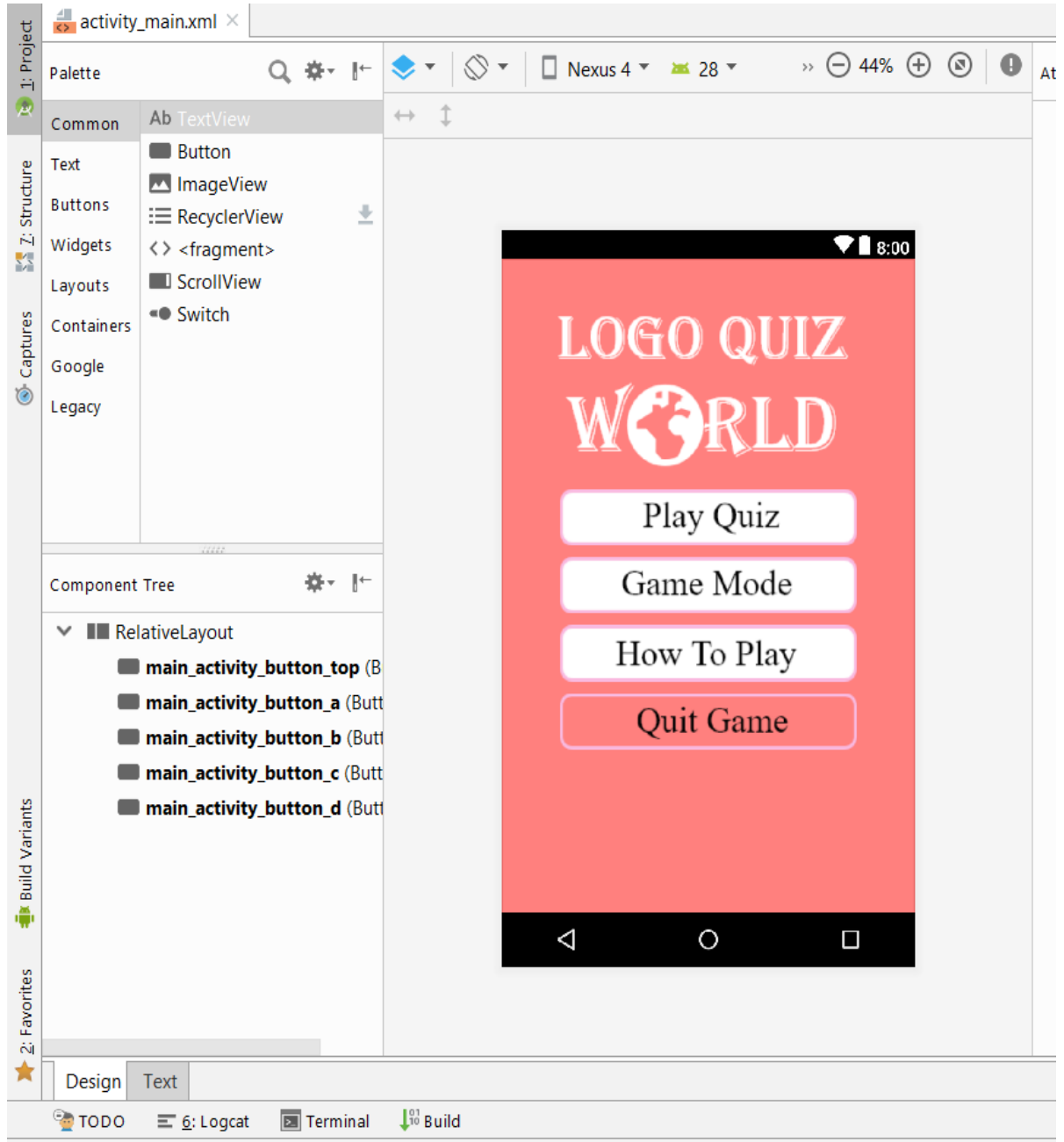
APPENDIX 1/2 Mock-Up Of “How To Play” Page



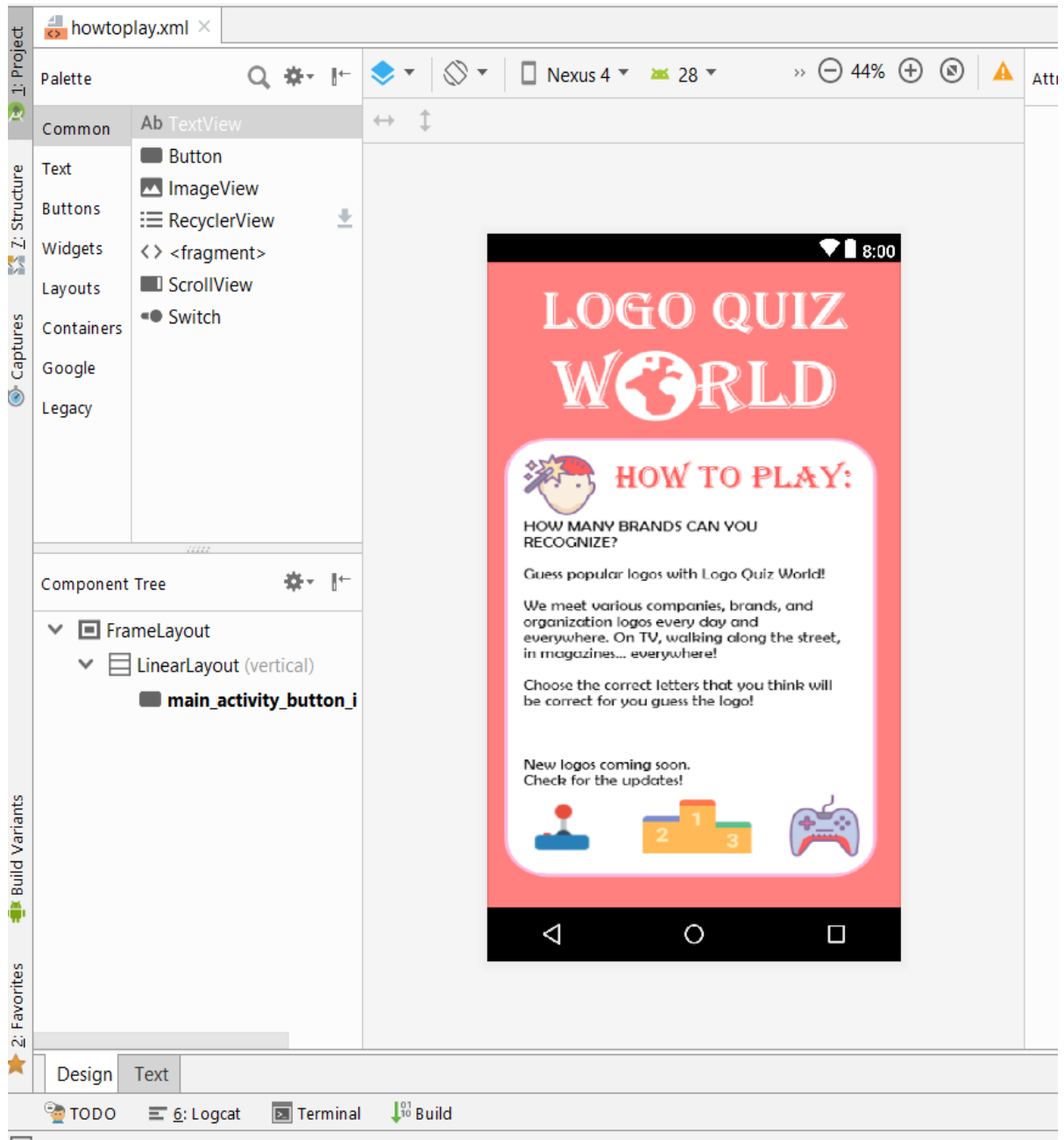
APPENDIX 1/3 The User Interface Of “Answering Quiz” Section.



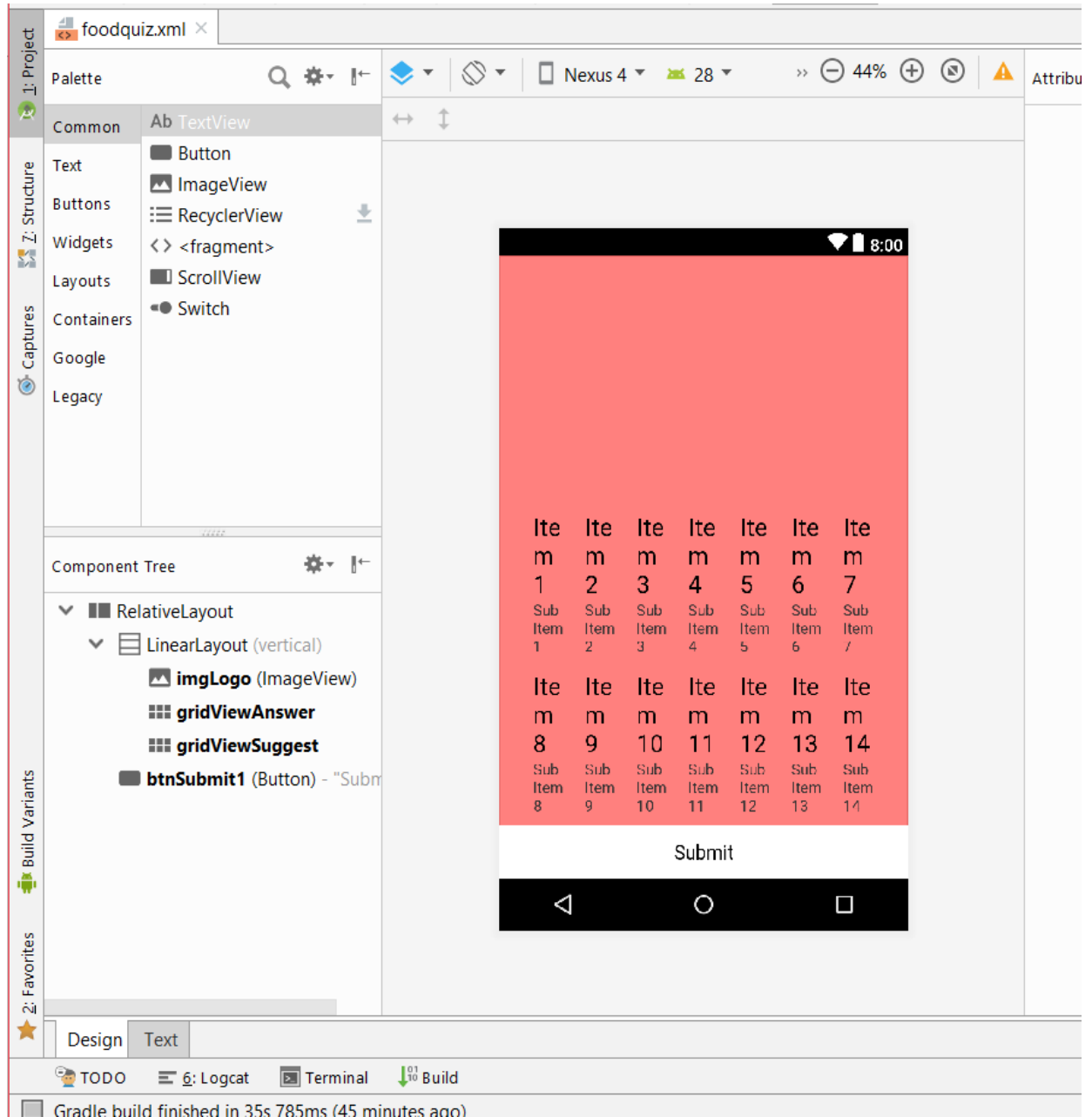
APPENDIX 2/1 “Main Screen” GUI



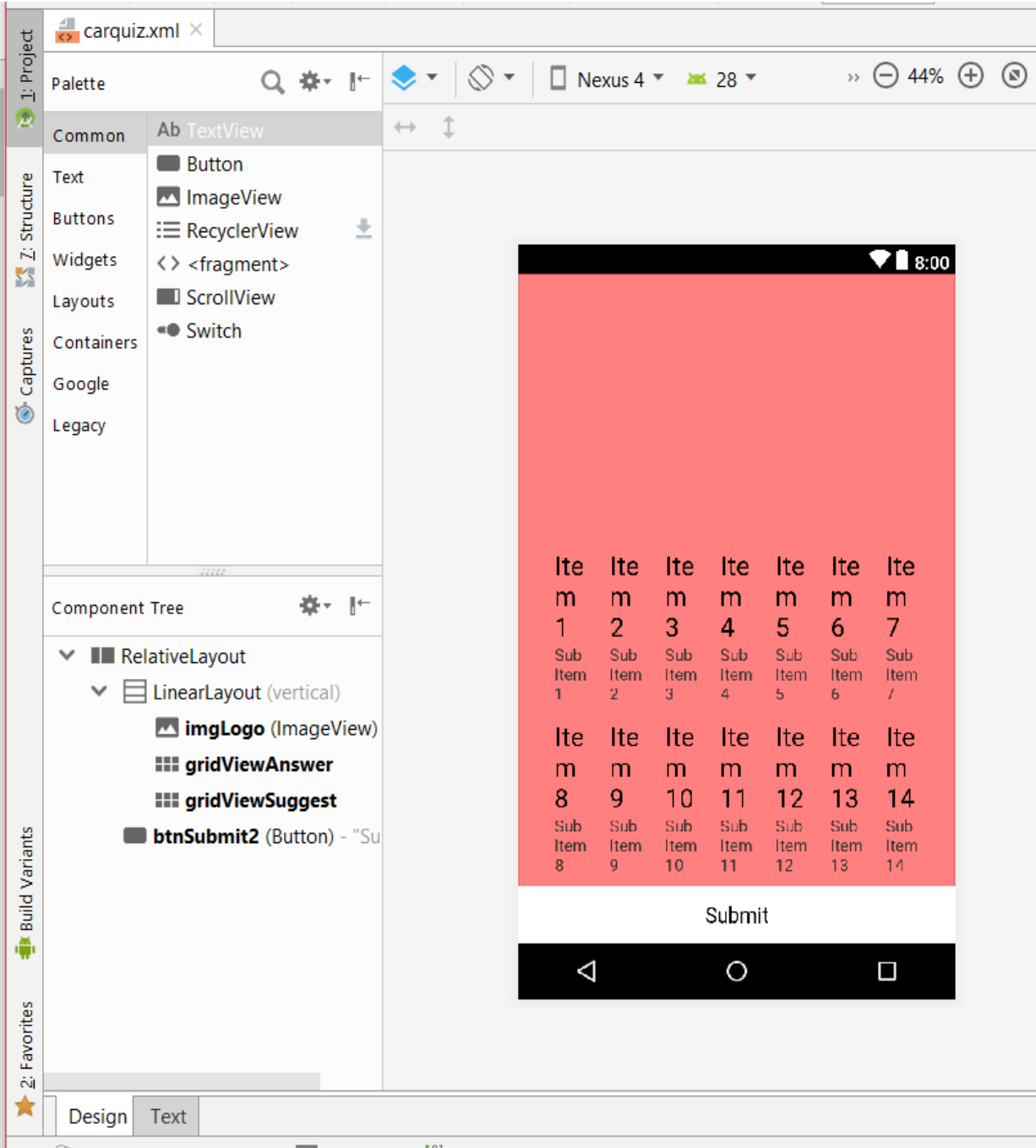
APPENDIX 2/2 “How To Play” GUI



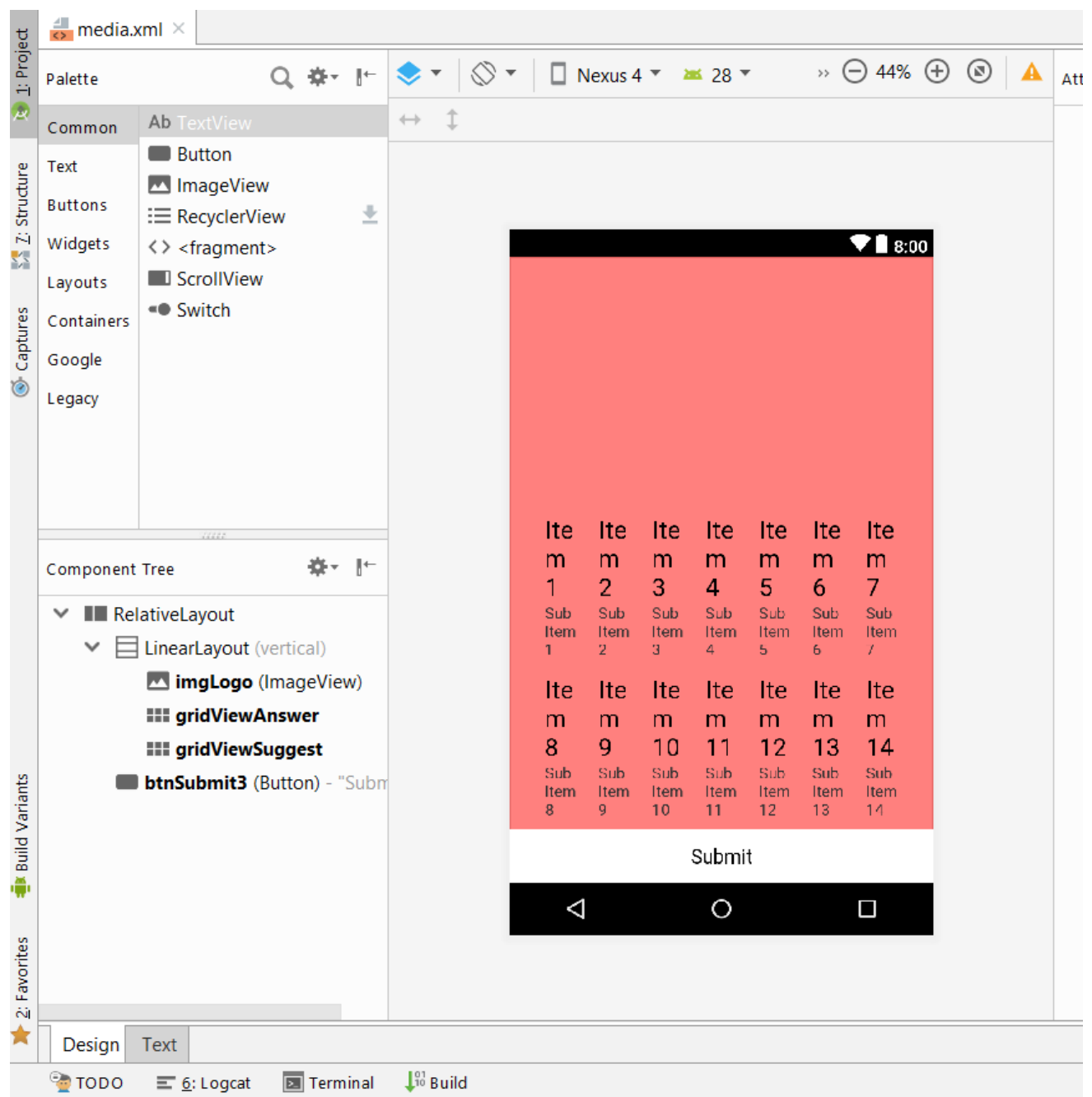
APPENDIX 2/3 “Food Quiz” GUI



APPENDIX 2/4 “Car Quiz” GUI



APPENDIX 2/5 “Social Media Quiz” GUI



APPENDIX 3/1 Main Activity Code (continue)

```
@Override
public void onClick(View v) {
    switch (v.getId()) {
        case R.id.main_activity_button_a: {
            Intent i = new Intent( packageContext: MainActivity.this, ActivityA.class);
            startActivity(i);
        } break;

        case R.id.main_activity_button_b: {
            Intent i = new Intent( packageContext: MainActivity.this, GameMode.class);
            startActivity(i);
        } break;

        case R.id.main_activity_button_c: {
            Intent i = new Intent( packageContext: MainActivity.this, Howtoplay.class);
            startActivity(i);
        } break;

        case R.id.main_activity_button_d: {
            finish();
        } break;

        default: Log.e(TAG, msg: "unexpected ID encountered");
    }
}
```


APPENDIX 3/2 Activity A Code (continue)

```

        public char[] answer;

        String correct_answer;

        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_a);
            /*
            Button b = null;
            b = (Button) findViewById(R.id.main_activity_butto
            b.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    finish();
                }
            });
            */
            initView();
        }
    }

```

```

private void initView() {
    gridViewAnswer = (GridView) findViewById(R.id.gridViewAnswer);
    gridViewSuggest = (GridView) findViewById(R.id.gridViewSuggest);

    imageViewQuestion = (ImageView) findViewById(R.id.imgLogo);

    //Add SetupList Here
    setupList();

    btnSubmit = (Button) findViewById(R.id.btnSubmit);
    btnSubmit.setOnClickListener((v) -> {
        String result="";
        for(int i=0;i< Common.user_submit_answer.length;i++)
            result+=String.valueOf(Common.user_submit_answer[i]);
        if(result.equals(correct_answer))
        {
            Toast.makeText(getApplicationContext(), text: "Good Job ! This is "+result, Toast.LENGTH_SHORT).show();

            //Reset
            Common.count = 0;
            Common.user_submit_answer = new char[correct_answer.length()];

            //Set Adapter
            GridViewAnswerAdapter answerAdapter = new GridViewAnswerAdapter(setupNullList(), getApplicationContext());
            gridViewAnswer.setAdapter(answerAdapter);
            answerAdapter.notifyDataSetChanged();

            GridViewSuggestAdapter suggestAdapter = new GridViewSuggestAdapter(suggestSource, getApplicationContext(), activityA: ActivityA.this);
            gridViewSuggest.setAdapter(suggestAdapter);
            suggestAdapter.notifyDataSetChanged();
        }
    });
}

```

```

        //Set Adapter
        GridViewAnswerAdapter answerAdapter = new GridViewAnswerAdapter(setupNullList(),getApplicationContext());
        gridViewAnswer.setAdapter(answerAdapter);
        answerAdapter.notifyDataSetChanged();

        GridViewSuggestAdapter suggestAdapter = new GridViewSuggestAdapter(suggestSource,getApplicationContext(), activity: ActivityA.this);
        gridViewSuggest.setAdapter(suggestAdapter);
        suggestAdapter.notifyDataSetChanged();

        setupList();
    }
    else
    {
        Toast.makeText(context: ActivityA.this, text: "Ops! Please Try Again!", Toast.LENGTH_SHORT).show();
    }
});

```

```

private void setupList() {
    //Random logo
    Random random = new Random();
    int imageSelected = image_list[random.nextInt(image_list.length)];
    imageViewQuestion.setImageResource(imageSelected);

    correct_answer = getResources().getResourceName(imageSelected);
    correct_answer = correct_answer.substring(correct_answer.lastIndexOf("/") + 1);

    answer = correct_answer.toCharArray();

    Common.user_submit_answer = new char[answer.length];

    //Add Answer character to List
    suggestSource.clear();
    for(char item:answer)
    {
        //Add logo name to list
        suggestSource.add(String.valueOf(item));
    }

    //Random add some character to list
    for(int i = answer.length; i < answer.length * 2; i++)
        suggestSource.add(Common.alphabet_character[random.nextInt(Common.alphabet_character.length)]);

    //Sort random
    Collections.shuffle(suggestSource);
}

```

```

182         suggestSource.add(String.valueOf(item));
183     }
184
185     //Random add some character to list
186     for(int i = answer.length;i<answer.length*2;i++)
187         suggestSource.add(Common.alphabet_character[random.nextInt(Common.alphabet_character.length)]);
188
189     //Sort random
190     Collections.shuffle(suggestSource);
191
192     //Set for GridView
193     answerAdapter = new GridViewAnswerAdapter(setupNullList(), context: this);
194     suggestAdapter = new GridViewSuggestAdapter(suggestSource, context: this, activityA: this);
195
196     answerAdapter.notifyDataSetChanged();
197     suggestAdapter.notifyDataSetChanged();
198
199     gridViewSuggest.setAdapter(suggestAdapter);
200     gridViewAnswer.setAdapter(answerAdapter);
201
202
203 }
204
205 @
206 private char[] setupNullList() {
207     char result[] = new char[answer.length];
208     for(int i=0;i<answer.length;i++)
209         result[i]=' ';
210     return result;
211 }

```

APPENDIX 3/3 Game Mode Code(continue)



```
28
29
30  @Override
31  public void onClick(View v) {
32      switch (v.getId()) {
33          case R.id.main_activity_button_f: {
34              Intent i = new Intent( packageContext: GameMode.this, Foodquiz.class);
35              startActivity(i);
36
37          }break;
38
39          case R.id.main_activity_button_g: {
40              Intent i = new Intent( packageContext: GameMode.this, CarQuiz.class);
41              startActivity(i);
42
43          }break;
44
45
46          case R.id.main_activity_button_h: {
47              Intent i = new Intent( packageContext: GameMode.this, Media.class);
48              startActivity(i);
49
50          }break;
51      }
52  }
53
54
```

APPENDIX 3/4 Food Quiz Code

```
public class Foodquiz extends Activity {
    public List<String> suggestSource1 = new ArrayList<>();

    public GridViewAnswerAdapterFood answerAdapter1;
    public GridViewSuggestAdapterFood suggestAdapter1;

    public Button btnSubmit1;

    public GridView gridViewAnswer1, gridViewSuggest1;

    public ImageView imgViewQuestion1;

    int[] image_list1={

        R.drawable.burgerking,
        R.drawable.kfc,
        R.drawable.wendys,
        R.drawable.mcdonalds,
        R.drawable.mountaindew,
        R.drawable.pizzahut,
        R.drawable.nestle,
        R.drawable.starbucks,
        R.drawable.knorr,
        R.drawable.pringles

    };
};
```

```
public char[] answer1;

String correct_answer1;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.foodquiz);
    /*
    Button b = null;
    b = (Button) findViewById(R.id.main_activity_button_a);
    b.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            finish();
        }
    });
    */
    initView();
}
```

```

private void initView() {
    gridViewAnswer1 = (GridView) findViewById(R.id.gridViewAnswer);
    gridViewSuggest1 = (GridView) findViewById(R.id.gridViewSuggest);

    imgViewQuestion1 = (ImageView) findViewById(R.id.imgLogo);

    //Add SetupList Here
    setupList();

    btnSubmit1 = (Button) findViewById(R.id.btnSubmit1);
    btnSubmit1.setOnClickListener((v) -> {
        String result="";
        for(int i=0;i< Common.user_submit_answer.length;i++)
            result+=String.valueOf(Common.user_submit_answer[i]);
        if(result.equals(correct_answer1))
        {
            Toast.makeText(getApplicationContext(), text: "Good Job ! This is "+result, Toast.LENGTH_SHORT).show();

            //Reset
            Common.count = 0;
            Common.user_submit_answer = new char[correct_answer1.length()];

            //Set Adapter
            GridViewAnswerAdapterFood answerAdapter = new GridViewAnswerAdapterFood(setupNullList(),getApplicationContext());
            gridViewAnswer1.setAdapter(answerAdapter);
            answerAdapter.notifyDataSetChanged();

            GridViewSuggestAdapterFood suggestAdapter = new GridViewSuggestAdapterFood(suggestSource1,getApplicationContext(), foodquiz: Food
            gridViewSuggest1.setAdapter(suggestAdapter);
            suggestAdapter.notifyDataSetChanged();

```

```

            //Set Adapter
            GridViewAnswerAdapterFood answerAdapter = new GridViewAnswerAdapterFood(setupNullList(),getApplicationContext());
            gridViewAnswer1.setAdapter(answerAdapter);
            answerAdapter.notifyDataSetChanged();

            GridViewSuggestAdapterFood suggestAdapter = new GridViewSuggestAdapterFood(suggestSource1,getApplicationContext(), foodquiz: Food
            gridViewSuggest1.setAdapter(suggestAdapter);
            suggestAdapter.notifyDataSetChanged();

            setupList();
        }
        else
        {
            Toast.makeText(context: Foodquiz.this, text: "Ops! Please Try Again!", Toast.LENGTH_SHORT).show();
        }
    });
}

```

```

private void setupList() {
    //Random logo
    Random random = new Random();
    int imageSelected = image_list1[random.nextInt(image_list1.length)];
    imageViewQuestion1.setImageResource(imageSelected);

    correct_answer1 = getResources().getResourceName(imageSelected);
    correct_answer1 = correct_answer1.substring(correct_answer1.lastIndexOf("/") + 1);

    answer1 = correct_answer1.toCharArray();

    Common.user_submit_answer = new char[answer1.length];

    //Add Answer character to List
    suggestSource1.clear();
    for(char item:answer1)
    {
        //Add logo name to list
        suggestSource1.add(String.valueOf(item));
    }

    //Random add some character to list
    for(int i = answer1.length; i < answer1.length * 2; i++)
        suggestSource1.add(Common.alphabet_character[random.nextInt(Common.alphabet_character.length)]);

    //Sort random
    Collections.shuffle(suggestSource1);

```

```

        //Sort random
        Collections.shuffle(suggestSource1);

        //Set for GridView
        answerAdapter1 = new GridViewAnswerAdapterFood(setupNullList(), context: this);
        suggestAdapter1 = new GridViewSuggestAdapterFood(suggestSource1, context: this, foodquiz: this);

        answerAdapter1.notifyDataSetChanged();
        suggestAdapter1.notifyDataSetChanged();

        gridViewSuggest1.setAdapter(suggestAdapter1);
        gridViewAnswer1.setAdapter(answerAdapter1);
    }

    private char[] setupNullList() {
        char result[] = new char[answer1.length];
        for(int i=0; i<answer1.length; i++)
            result[i]=' ';
        return result;
    }
}

```

APPENDIX 3/5 Car Quiz Code

```
public class CarQuiz extends Activity {
    public List<String> suggestSource2 = new ArrayList<>();

    public GridViewAnswerAdapterCar answerAdapter2;
    public GridViewSuggestAdapterCar suggestAdapter2;

    public Button btnSubmit2;

    public GridView gridViewAnswer2, gridViewSuggest2;

    public ImageView imgViewQuestion2;

    int[] image_list2={

        R.drawable.alfaromeo,
        R.drawable.audi,
        R.drawable.bmw,
        R.drawable.chevrolet,
        R.drawable.citroen,
        R.drawable.ferrari,
        R.drawable.honda,
        R.drawable.hyundai,
        R.drawable.jaguar,
        R.drawable.lexus,
        R.drawable.mazda,
        R.drawable.mercedesbenz,
        R.drawable.mitsubishi,

    }

    public char[] answer2;

    String correct_answer2;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.carquiz);
        /*
        Button b = null;
        b = (Button) findViewById(R.id.main_activity_button_a);
        b.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                finish();
            }
        });
        */
        initView();
    }
}
```



```

private void initView() {
    gridViewAnswer2 = (GridView)findViewById(R.id.gridViewAnswer);
    gridViewSuggest2 = (GridView)findViewById(R.id.gridViewSuggest);

    imageViewQuestion2 = (ImageView)findViewById(R.id.imgLogo);

    //Add SetupList Here
    setupList();

    btnSubmit2 = (Button)findViewById(R.id.btnSubmit2);
    btnSubmit2.setOnClickListener((v) -> {
        String result="";
        for(int i=0;i< Common.user_submit_answer.length;i++)
            result+=String.valueOf(Common.user_submit_answer[i]);
        if(result.equals(correct_answer2))
        {
            Toast.makeText(getApplicationContext(), text: "Good Job ! This is " +result, Toast.LENGTH_SHORT).show();

            //Reset
            Common.count = 0;
            Common.user_submit_answer = new char[correct_answer2.length()];

            //Set Adapter
            GridViewAnswerAdapterCar answerAdapter = new GridViewAnswerAdapterCar(setupNullList(),getApplicationContext());
            gridViewAnswer2.setAdapter(answerAdapter);
            answerAdapter.notifyDataSetChanged();

            GridViewSuggestAdapterCar suggestAdapter = new GridViewSuggestAdapterCar(suggestSource2,getApplicationContext(), carquiz: CarQuiz.this);
            gridViewSuggest2.setAdapter(suggestAdapter);

            GridViewSuggestAdapterCar suggestAdapter = new GridViewSuggestAdapterCar(suggestSource2,getApplicationContext(), carquiz: CarQuiz.this);
            gridViewSuggest2.setAdapter(suggestAdapter);
            suggestAdapter.notifyDataSetChanged();

            setupList();
        }
        else
        {
            Toast.makeText(context: CarQuiz.this, text: "Ops! Please Try Again!", Toast.LENGTH_SHORT).show();
        }
    });
}

private void setupList() {
    //Random logo
    Random random = new Random();
    int imageSelected = image_list2[random.nextInt(image_list2.length)];
    imageViewQuestion2.setImageResource(imageSelected);

    correct_answer2 = getResources().getResourceName(imageSelected);
    correct_answer2 = correct_answer2.substring(correct_answer2.lastIndexOf("/") +1);

    answer2 = correct_answer2.toCharArray();

    Common.user_submit_answer = new char[answer2.length];

```

```

//Add Answer character to List
suggestSource2.clear();
for(char item:answer2)
{
    //Add logo name to list
    suggestSource2.add(String.valueOf(item));
}
//Random add some character to list
for(int i = answer2.length;i<answer2.length*2;i++)
    suggestSource2.add(Common.alphabet_character[random.nextInt(Common.alphabet_character.length)]);
//Sort random
Collections.shuffle(suggestSource2);
//Set for GridView
answerAdapter2 = new GridViewAnswerAdapterCar(setupNullList(), context: this);
suggestAdapter2 = new GridViewSuggestAdapterCar(suggestSource2, context: this, carquiz: this);

answerAdapter2.notifyDataSetChanged();
suggestAdapter2.notifyDataSetChanged();

gridViewSuggest2.setAdapter(suggestAdapter2);
gridViewAnswer2.setAdapter(answerAdapter2);
}

private char[] setupNullList() {
    char result[] = new char[answer2.length];
    for(int i=0;i<answer2.length;i++)
        result[i]=' ';
    return result;
}
}

```

APPENDIX 3/6 Social Media Quiz Code

```
activity_a.xml × Media.java ×
1 package at.fhooe.mc.android;
2
3 import ...
19
20 public class Media extends Activity {
21     public List<String> suggestSource3 = new ArrayList<>();
22
23     public GridViewAnswerAdapterMedia answerAdapter3;
24     public GridViewSuggestAdapterMedia suggestAdapter3;
25
26     public Button btnSubmit3;
27
28     public GridView gridViewAnswer3, gridViewSuggest3;
29
30     public ImageView imgViewQuestion3;
31
32     int[] image_list3={
33
34
35         R.drawable.apple,
36         R.drawable.blogger,
37         R.drawable.deviantart,
38         R.drawable.digg,
39         R.drawable.dropbox,
40         R.drawable.evernote,
41         R.drawable.firefox,
42         R.drawable.flickr,
43         R.drawable.google,
44         R.drawable.googleplus,
```

```
public char[] answer3;

String correct_answer3;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.media);

    initView();
}

private void initView() {
    gridViewAnswer3 = (GridView) findViewById(R.id.gridViewAnswer);
    gridViewSuggest3 = (GridView) findViewById(R.id.gridViewSuggest);

    imgViewQuestion3 = (ImageView) findViewById(R.id.imgLogo);
```

```
activity_axml x Media.java x
88 //Add SetupList Here
89 setupList();
90 btnSubmit3 = (Button)findViewById(R.id.btnSubmit3);
91 btnSubmit3.setOnClickListener((v) -> {
92     String result="";
93     for(int i=0;i< Common.user_submit_answer.length;i++)
94         result+=String.valueOf(Common.user_submit_answer[i]);
95     if(result.equals(correct_answer3))
96     {
97         Toast.makeText(getApplicationContext(), text: "Good Job ! This is "+result, Toast.LENGTH_SHORT).show();
98
99         //Reset
100         Common.count = 0;
101         Common.user_submit_answer = new char[correct_answer3.length()];
102
103         //Set Adapter
104         GridViewAnswerAdapterMedia answerAdapter = new GridViewAnswerAdapterMedia(setupNullList(),getApplicationContext());
105         gridViewAnswer3.setAdapter(answerAdapter);
106         answerAdapter.notifyDataSetChanged();
107
108         GridViewSuggestAdapterMedia suggestAdapter = new GridViewSuggestAdapterMedia(suggestSource3,getApplicationContext(), media: Media.this)
109         gridViewSuggest3.setAdapter(suggestAdapter);
110         suggestAdapter.notifyDataSetChanged();
111         setupList();
112     }
113     else
114     {
115         Toast.makeText(context: Media.this, text: "Ops! Please Try Again!", Toast.LENGTH_SHORT).show();
116     }
117 }
118
```

```
activity_axml x Media.java x
122 private void setupList() {
123     //Random logo
124     Random random = new Random();
125     int imageSelected = image_list3[random.nextInt(image_list3.length)];
126     imageViewQuestion3.setImageResource(imageSelected);
127
128     correct_answer3 = getResources().getResourceName(imageSelected);
129     correct_answer3 = correct_answer3.substring(correct_answer3.lastIndexOf(str: "/")+1);
130
131     answer3 = correct_answer3.toCharArray();
132
133     Common.user_submit_answer = new char[answer3.length];
134
135     //Add Answer character to List
136     suggestSource3.clear();
137     for(char item:answer3)
138     {
139         //Add logo name to list
140         suggestSource3.add(String.valueOf(item));
141     }
142
143     //Random add some character to list
144     for(int i = answer3.length;i<answer3.length*2;i++)
145         suggestSource3.add(Common.alphabet_character[random.nextInt(Common.alphabet_character.length)]);
146
147     //Sort random
148     Collections.shuffle(suggestSource3);
149 }
150
```

```
activity.xml x Media.java x
146         suggestSource3.add(Common.alphabet_character[random.nextInt(Common.alphabet_character.length)]);
147
148         //Sort random
149         Collections.shuffle(suggestSource3);
150
151         //Set for GridView
152         answerAdapter3 = new GridViewAnswerAdapterMedia(setupNullList(), context: this);
153         suggestAdapter3 = new GridViewSuggestAdapterMedia(suggestSource3, context: this, media: this);
154
155         answerAdapter3.notifyDataSetChanged();
156         suggestAdapter3.notifyDataSetChanged();
157
158         gridViewSuggest3.setAdapter(suggestAdapter3);
159         gridViewAnswer3.setAdapter(answerAdapter3);
160
161
162     }
163
164     @ private char[] setupNullList() {
165         char result[] = new char[answer3.length];
166         for(int i=0;i<answer3.length;i++)
167             result[i]=' ';
168         return result;
169     }
170 }
171
```

APPENDIX 4/1 Testing With Genymotion

