|  |
| --- |
| **Is Android getting more popular?** |

Table of Contents

[Section A: Getting Started with Android 1](#_Toc38225233)

[Section B: App Development and Mobile App Ecosystem 2](#_Toc38225234)

[Section C: Create a Simple User Interface Layout 5](#_Toc38225235)

[Handling the Problem Statement 14](#_Toc38225236)

## Section A: Getting Started with Android

Refer to "**Getting Started with Android**" guide and ensure that you have the necessary setup done.

* Android Studio is installed
* An emulator is created
* Able to create a "Hello World" project and run the app using an emulator

You can also refer to "Getting Started with Android.pdf" in your Student Resources folder.

**Note:** As installation may take some time, you should proceed to do the rest of the worksheet concurrently.

Check out the following LinkedIn video [Connect a physical device for testing](https://www.linkedin.com/learning/android-studio-essential-training-2020/connect-a-physical-device-for-testing) to connect to a physical device for testing.

## Section B: App Development and Mobile App Ecosystem

1. Read the following articles
   1. <https://www.360logica.com/blog/mobile-apps-vs-desktop-apps-a-deeper-look/>
   2. <https://learntocodewith.me/posts/cross-platform-apps/>
   3. <https://www.webfx.com/blog/web-design/native-app-vs-mobile-web-app-comparison/>
2. Applications can be developed for desktop, web and mobile platform. What are the main considerations when developing applications for each of the platform?

|  |  |  |  |
| --- | --- | --- | --- |
| **Consideration** | **Desktop**  **(e.g. Word)** | **Web**  **(e.g. Facebook on IE)** | **Mobile**  **(e.g. WhatsApp)** |
| **Size of Screen** | Desktop usually has a larger screen to manipulate with. | Desktop and Web have a larger display size. | Smaller display size although it is getting bigger. |
| **Connectivity**  **(Need Internet?)** | Offline and Online | Online | Online and offline |
| **Processing Power** | ‘Highest’ consumption of processing power | Lower than desktop, higher than mobile | Lower then web as its just an app for application. |
| **No. of users** | 1 user | Multi user access | 1 user |
| **Add-on applications** | Installation maybe required | Run directly on web browser | Can be download and installed from digital distribution platform |
| **Life Cycle** | Application runs till user ends it | Application runs till ‘session timeout’ | Application runs till user ends it or interruption occurs |
| **Availability** | Only available on the computer which the application is installed | available anywhere with any web browser as long as there is internet connection. | Only available on the mobile phone which the application is installed |

1. Based on your comparison above, what is the most suitable platform for the application to be developed in each scenario below?

|  |  |
| --- | --- |
| **Scenario** | **Platform** |
| No installation is required to use the application. | Web |
| Processing power (CPU) and memory (RAM) are very limited. | mobile |
| Screen has large display size and the app can be used when there is no Internet connection. | Desktop |

1. Compare the two major smartphone operating systems worldwide, Android and iOS, and fill in the table below.

(<https://www.idc.com/promo/smartphone-market-share/os>)

|  |  |  |
| --- | --- | --- |
|  | **iOS** | **Android** |
| **Owner** | Apple | Google |
| **Programming Language(s)** | Swift, Objective-C | Java, kotlin |
| **Programming Tool** | XCode | Android Studio |
| **Open Source?** | The vast majority of the OS is closed-source. | Android is an open source for majority a products |
| **Market Share (2020)** | 16.2% | 84.1% |

1. List down at least 5 phone models of at least 3 brands that are based on Android OS.

(<https://www.android.com/phones/>

<https://www.idc.com/promo/smartphone-market-share/vendor>)

|  |
| --- |
| Samsung:  Samsung S21  Samsung Note 20 ultra  Hwa wei:  Hwa wei Mate 20 Hwa wei P30  Google:  Google Pixel 4 |

1. Watch the following video in LinkedIn to find out more about Android history.

<https://www.linkedin.com/learning/android-development-essential-training-your-first-app-with-kotlin/explore-the-history-of-android?u=41614228>

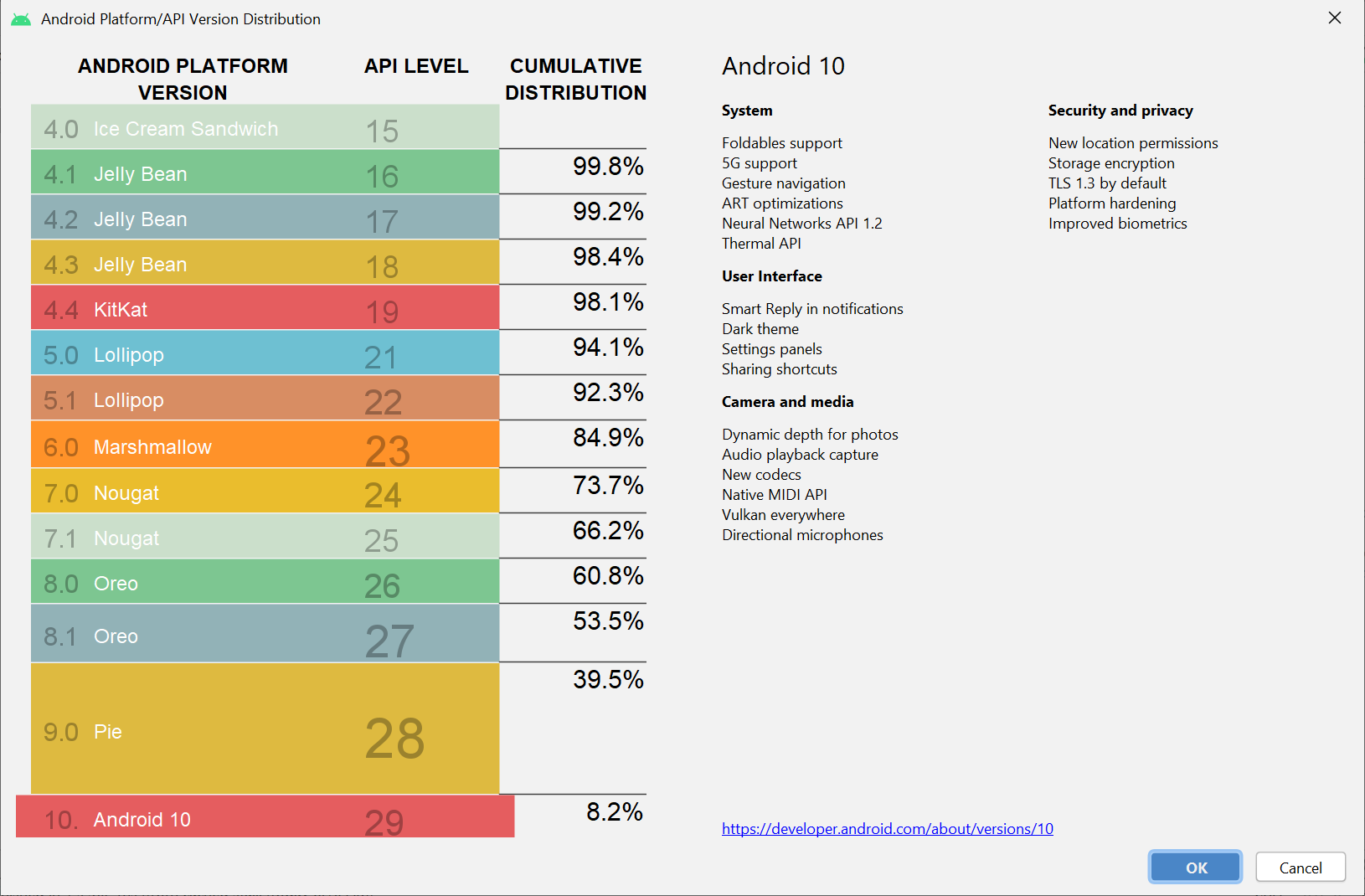
1. Each Android version equips with some new features/functions. E.g., NFC was implemented in Android 2.3 Gingerbread.

Refer to Android Versions for the list of Android versions and their corresponding new features/functions <https://www.javatpoint.com/android-versions>

Fill in the Android version and codename according to the feature/function added.

|  |  |
| --- | --- |
| **Feature/Function** | **Android Version and Codename** |
| Near Field Communication (NFC) | Android 2.3 Gingerbread |
| Material Design | 5.0 – 5.1.1 Lollipop |
| Permissions | 6.0 - 6.0.1: Marshmallow |
| Picture-in-Picture | 7.0 – 7.1.2: Nougat |
| Voice: OK Google | 4.4 – 4.4.4: Kitkat |
| Notifications | 1.0 – 1.1: No codename |

1. Based on the following Android Platform/API Version Distribution table, it shows you the cumulative distribution based on the API level selected. As an android developer, select the oldest version of Android that you would like your app to run on, and your app will run smoothly on all versions newer than that.



If the minimum API level chosen is 29, what’s the percentage of user base that your app will cater to?

|  |
| --- |
| 8.2% |

1. Since there are many different platform versions of Android OS in various handset model, as a developer, which platform versions should you develop on, so that most of the Android users can benefit from your application?

(Hint: <https://developer.android.com/training/basics/supporting-devices/platforms.html>)

|  |  |  |
| --- | --- | --- |
|  | **API Level** | **Platform Version** |
| Minimum SDK | 16 | 4.1 |
| Target SDK | 30 | 11 |

## Section C: Create a Simple User Interface Layout

**You can start this section when the installation has been done and your first "Hello World" app has been created and run, by following the "Getting Started with Android ".**

Android App User Interface is defined in the layout file which resides under res -> layout folder of the project. Firstly, let us learn how to add new UI elements, e.g., TextView (commonly known as label in the earlier C294 Mobile User Interface module), into the Android layout.

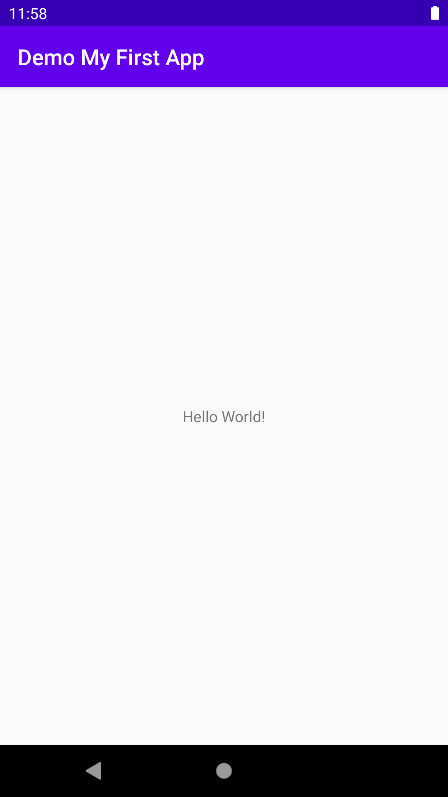
1. In Android Studio, create a new project with the following information:-

|  |  |
| --- | --- |
| **Project Template** | Empty Activity |
| **Application Name** | Demo My First App |
| **Package Name** | sg.edu.rp.c346.id<your student ID>. demomyfirstapp  E.g. sg.edu.rp.c346.id18123456. demomyfirstapp |
| **Project Location** | D:\C346\Workspace\P01\DemoMyFirstApp |
| **Language** | Java |
| **Minimum API Level** | API 16 |

1. There are 2 files created and opened in the main window. Fill in the table below with the correct filename.

|  |  |
| --- | --- |
| **Filename** | **Description** |
| Activity\_main.xml | A file used for designing layout |
| MainActivity.java | Java file that controls the behaviour of the screen/activity |

1. Run your app. Note that for a newly created project, "Hello World" will be shown on screen.

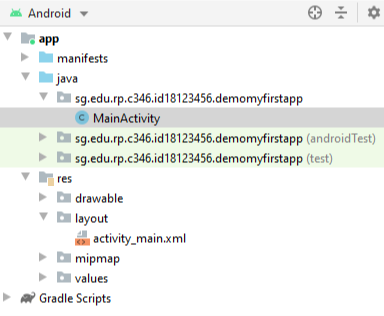


In which file can you find the text "Hello World!"?

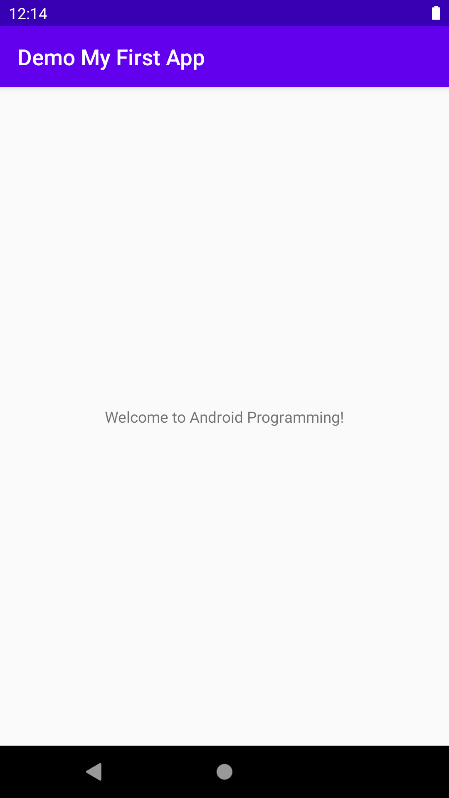
|  |
| --- |
| Activity\_main.xml would be the file would be the one that would allow us to see “Hello world” on the application. |

1. The left panel in Android Studio displays the files in Android Project Scope/View.

There are some folders in the project. In the proceeding sections, we will be focusing on one of them, the res (resource) folder, which contains resource files (layout files, images, etc.).



1. Make a change in the layout file to change the message as shown below.

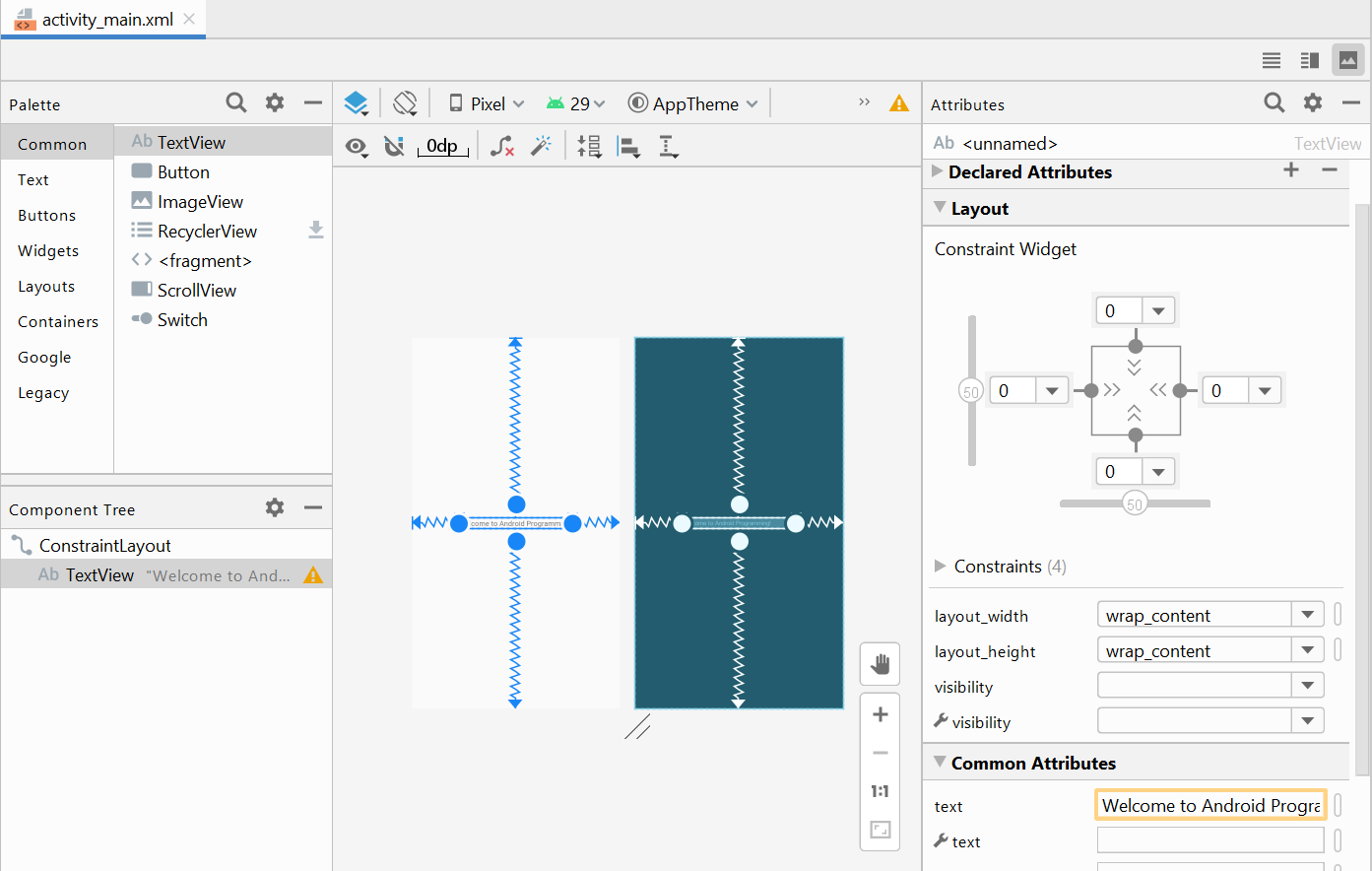


1. The default layout used when a project is first created is ConstraintLayout. In this module, we will be using LinearLayout, which is simpler to use. Let’s edit the layout file. From the "**Android**" scope shown in the left panel, open **res → layout → activity\_main.xml**.

**Design Mode**

The following screen shows the layout in **Design Mode**.

You can preview your layout in the preview screen.



**Preview Screen**

**Code Mode**

Select **Code Mode** to view the XML code.



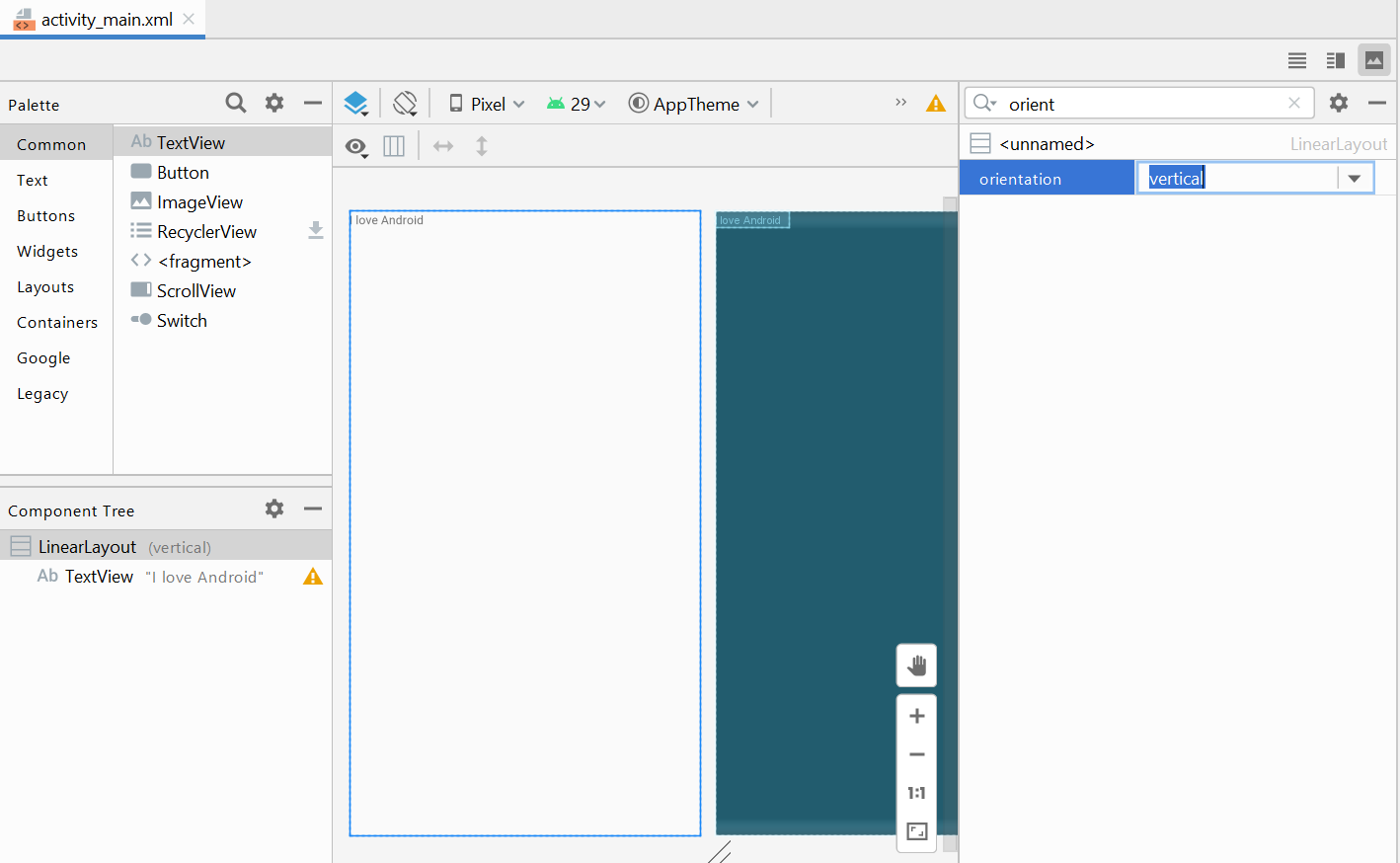
Replace the default codes with the following code:

|  |
| --- |
| *<?***xml version="1.0" encoding="utf-8"***?>* <**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:orientation="horizontal"**>   <**TextView  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="I love Android"** />  </**LinearLayout**> |

|  |  |
| --- | --- |
| ? | Examine the Preview screen. What did you see? |
| I see the text being on the top left corner. | |

1. Let’s change the orientation of the layout from the default horizontal to vertical so that we can arrange the controls vertically later.
   1. Select the LinearLayout (horizontal) under the Component Tree
   2. Under the Attributes tab on the right, change the orientation to vertical

* You may find the attribute in the search bar



* Alternatively, you can find it under **All Attributes** section.

|  |  |
| --- | --- |
|  |  |

1. In the Design screen, both design and blueprint are shown by default.
   1. Select  to see Design only.

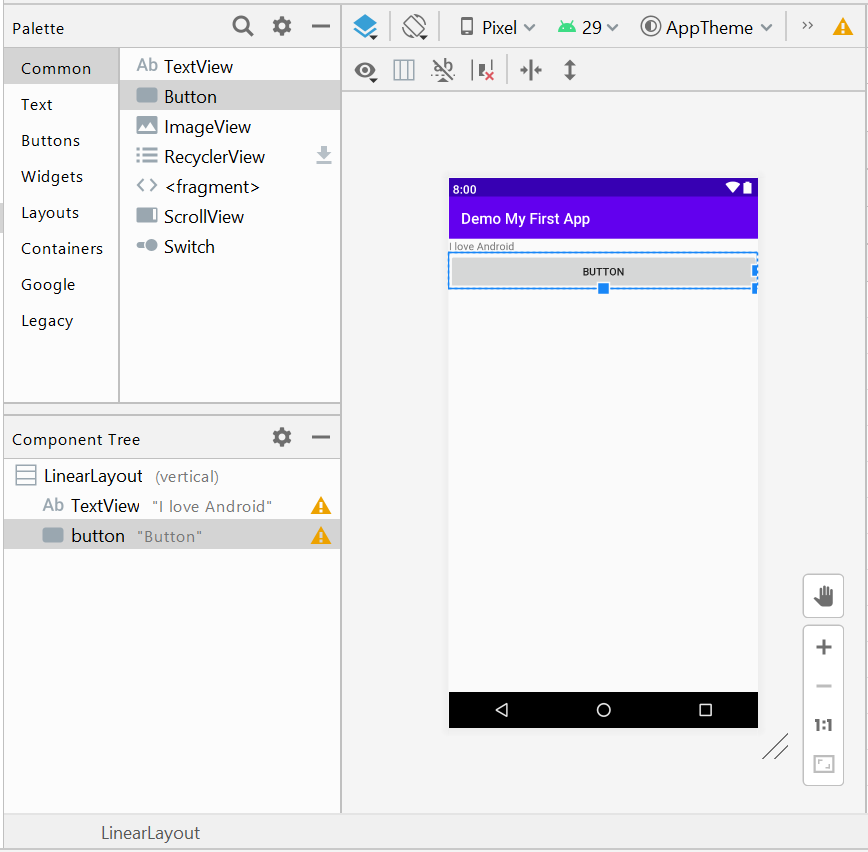
|  |  |
| --- | --- |
|  |  |

* 1. Choose  and select Show Layout Decorations. You’ll be able to the preview of the entire screen, including the app bar.

|  |  |
| --- | --- |
|  |  |

1. Let’s add some more UI element to the screen.
   1. Drag a Button control under Widgets category in the Palette
   2. Drop it below the text in the device screen

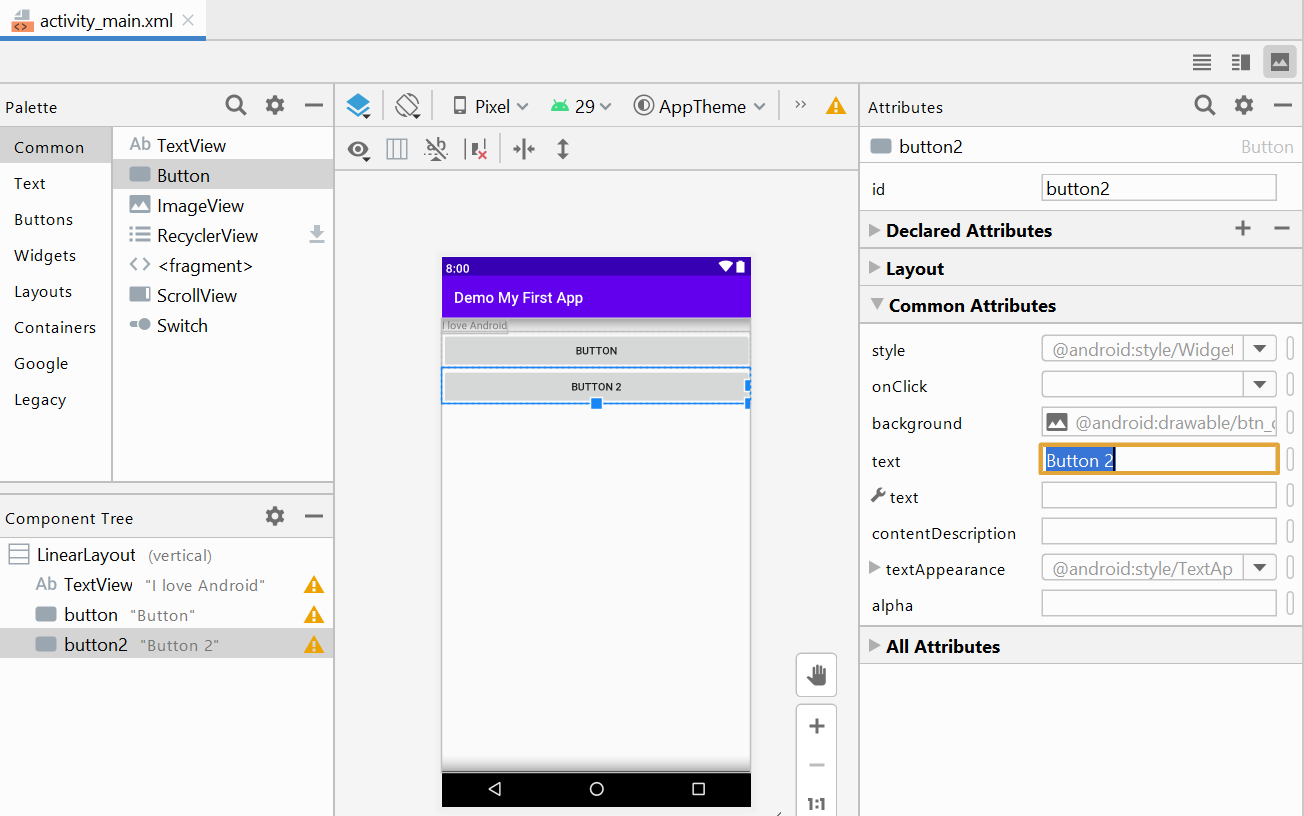
It should look like the screen below once this step is executed correctly.



|  |  |
| --- | --- |
| ? | Switch to Text Mode to view the XML code. Identify the relative location where the code for the **Button** is generated. |
| The location where the Button is generated would be in the activity\_main.xml | |

1. Place another button in the layout and update the text of the button to "Button 2".

You can drag a Button into the Component Tree.



You can move the various UI elements in the Component Tree.

Look at the XML code to see how the UI elements are defined.

|  |
| --- |
| *<?***xml version="1.0" encoding="utf-8"***?>* <**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:orientation="vertical"**>   <**TextView  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="I love Android"** />   <**Button  android:id="@+id/button"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="Button"** />   <**Button  android:id="@+id/button2"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="Button 2"** />  </**LinearLayout**> |

1. Update the LinearLayout’s orientation from **vertical** to **horizontal**.

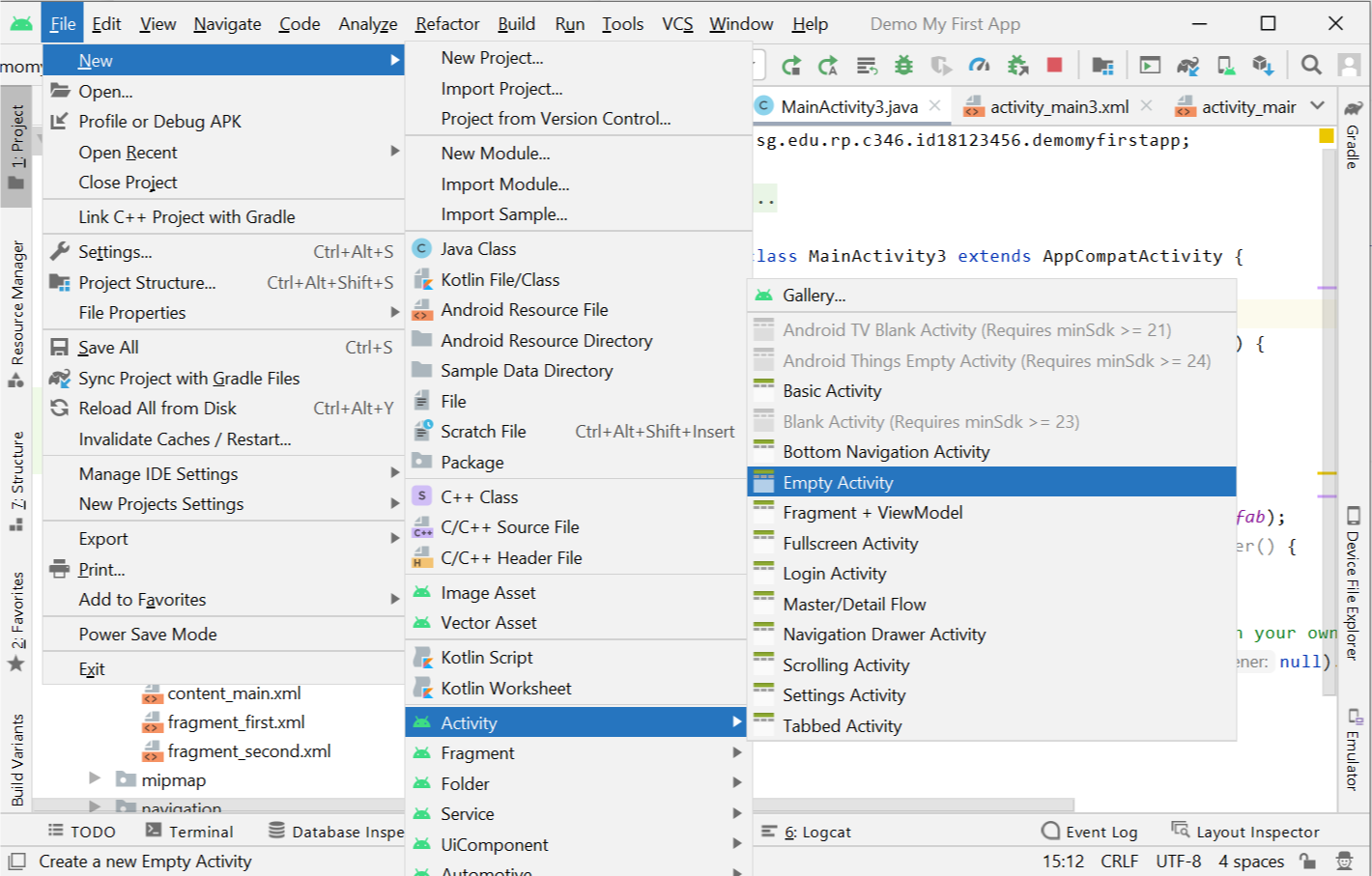
You can do so using the design mode or edit the XML codes directly.



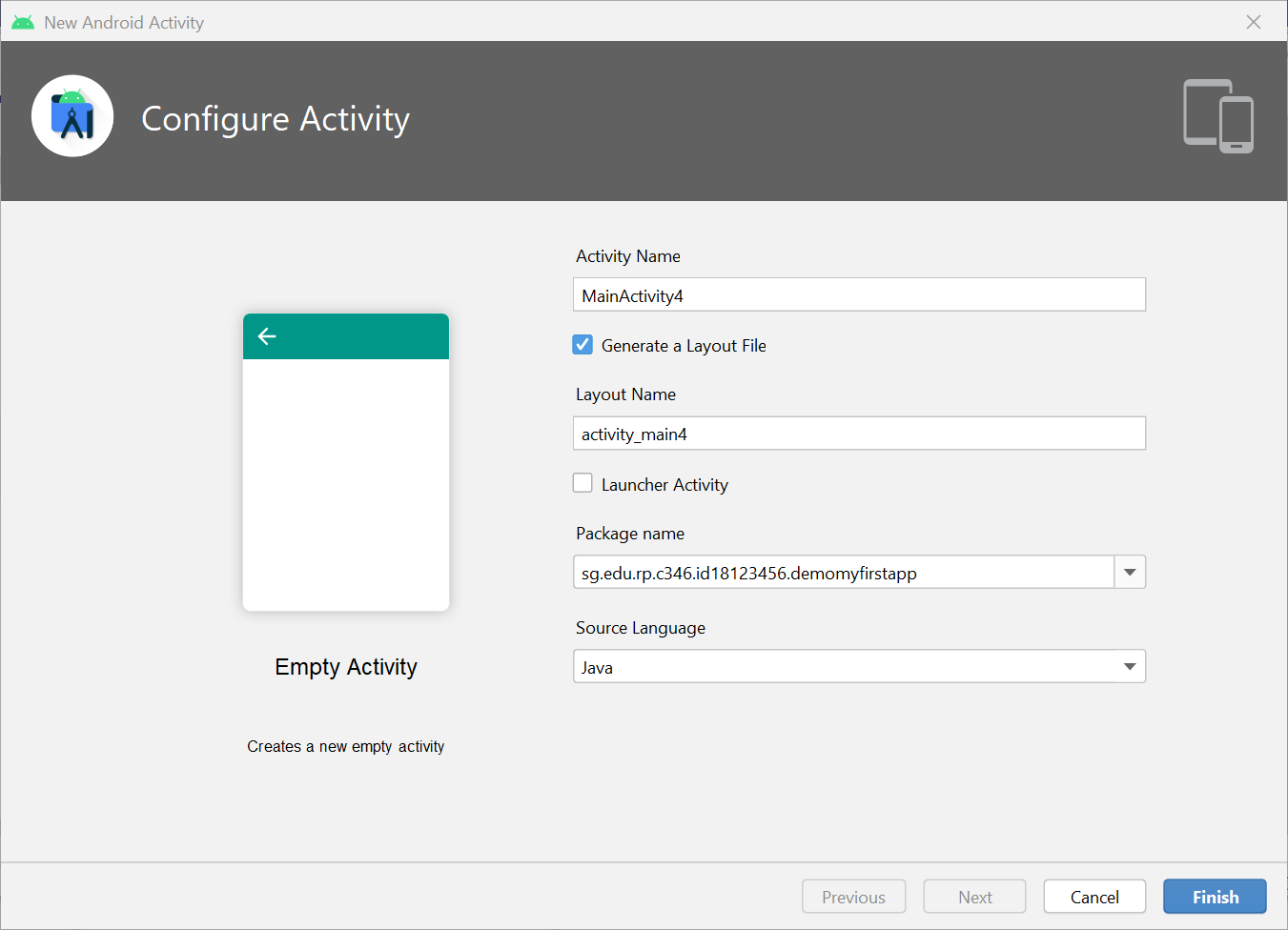
|  |  |
| --- | --- |
| ? | Any changes to the relative location of the **button** control now? Write the possible reason for the changes. |
| The changes that I was able to observe would be the button would be off screen and since the button is too big they would just be compressed since the screen is too small for it to fit both. | |

|  |  |
| --- | --- |
| ? | What happened to the second button?  Hint: Update the **layout\_width** attribute for the first Button from "**match\_parent**" to "**wrap\_content**". |
| If we would to update the button it would be able to see both the button now. | |

1. Create a new screen by creating a new activity. From the menu, click File > Activity > Empty Activity.



1. Accept the suggested Activity and Layout Name in the dialog that appears and click Finish.



1. You can modify both the XML and Java files like you had done for the first activity to build the UI for the new activity you had created.

|  |  |
| --- | --- |
|  | **Learning Checkpoint** |
| *By now, you should be able to*   * *Draw simple UI via the IDE designer tool in Design mode.* * *Make simple modification to the layout orientation by changing the layout attributes e.g. from android:orientation="vertical" to android:orientation="horizontal"*   **To recap on what we have learned so far,**  The designer tool provides a “what you see is what you get” (WYSIWYG) environment in which views can be selected from a palette and then placed onto a “Preview Screen” which is a canvas representing the display of an Android device.  The Android UI can be visually manipulated by directly working with the palette, while one could view the corresponding XML code and even edit it via the Text mode. | |

## Handling the Problem Statement

1. Propose an app you would like to create and provide the overview and functionality of your app.
2. Create one of the activities (screens) of the app.

A sample screen has been given for reference. Be creative and come out with your own design! You are encouraged to explore and use various User Interface (UI) components.

Create a new project for the problem statement with the following requirements:

|  |  |
| --- | --- |
| **Project Template** | Empty Activity |
| **Application Name** | Simple App |
| **Package Name** | sg.edu.rp.c346.id<your student ID>.simpleapp  E.g. sg.edu.rp.c346.id18123456.simpleapp |
| **Project Location** | D:\C346\Workspace\P01\SimpleApp |
| **Language** | Java |
| **Minimum API Level** | API 16 |

For the input text field, you may try to use the controls under Text, like Plain Text, Password, E-mail etc.

|  |  |
| --- | --- |
| C:\Users\denise_quek\Desktop\Screenshot_1586410555.png |  |

|  |
| --- |
| **Instructions**  Prepare presentation slides which you will use to show to the investors. You may choose to use Microsoft Word or PowerPoint. Place the write up and the screenshot of the activity in the slides.   * Name your document as "proposal.docx" / "proposal.pptx". |