# Name: Nguyễn Văn Cường – Student ID: 20215006

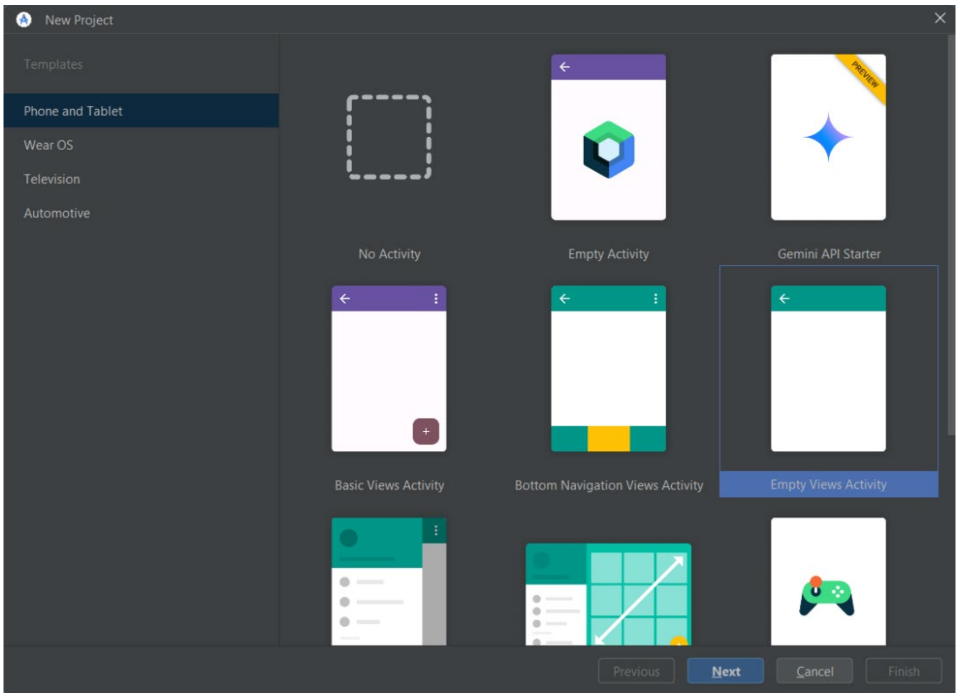
## Report – Lesson 4

## 4.1. Create a Birthday Card app

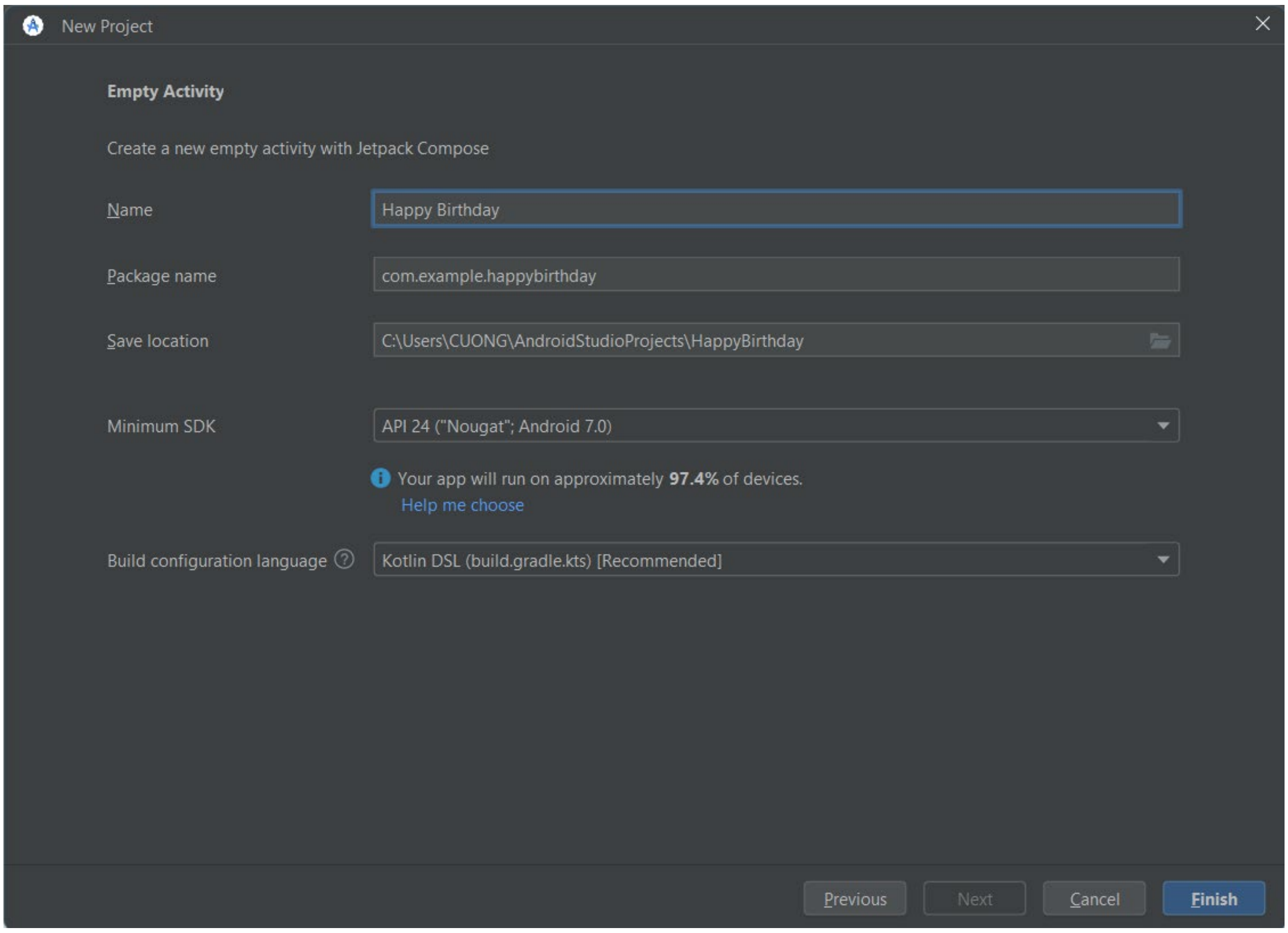
## Set up your Happy Birthday app

### Create an Empty Activity project

1. To start, create a new Kotlin project in Android Studio using the Empty Views Activity template.



2. Call the app "Happy Birthday", with a minimum API level of 24 (Nougat).


New Project

**Empty Activity**

Create a new empty activity with Jetpack Compose

Name: Happy Birthday

Package name: com.example.happybirthday

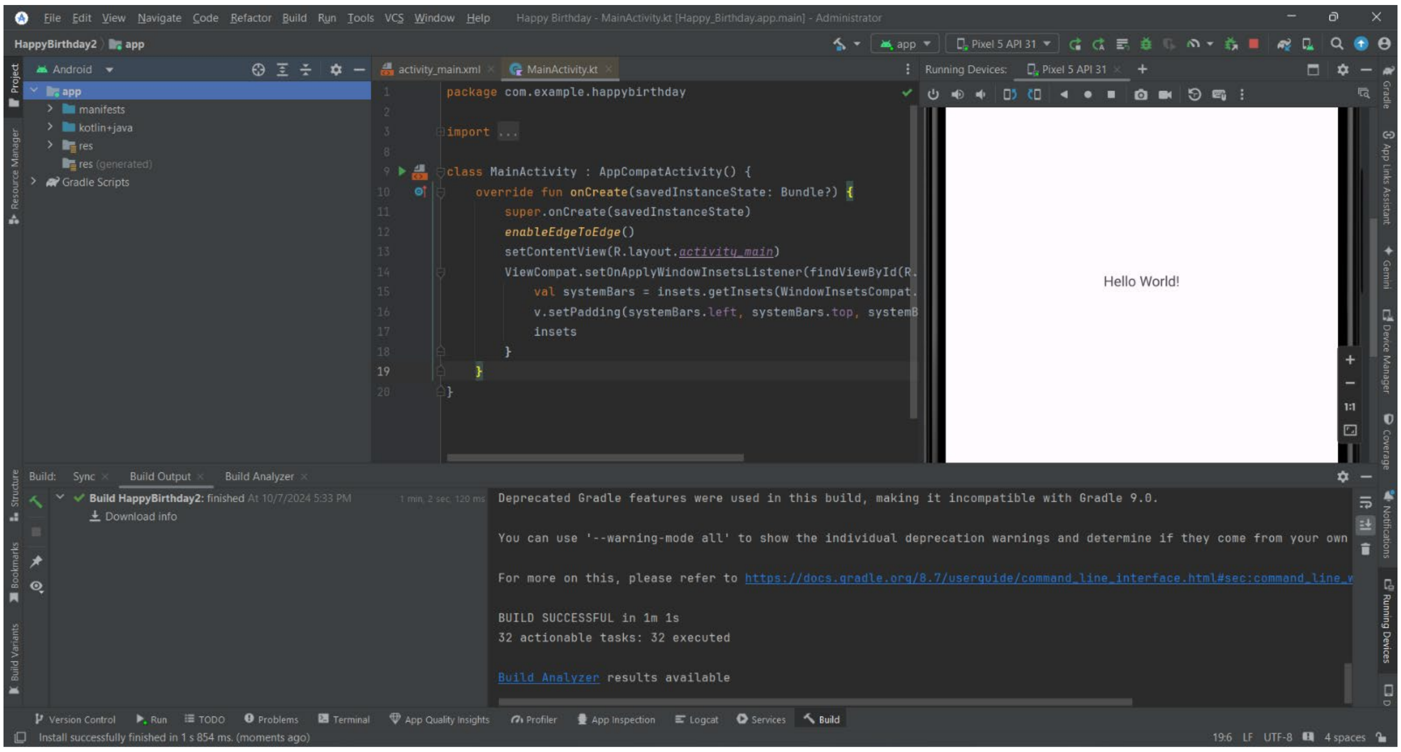
Save location: C:\Users\CUONG\AndroidStudioProjects\HappyBirthday

Minimum SDK: API 24 ("Nougat"; Android 7.0)

Your app will run on approximately **97.4%** of devices.
Help me choose

Build configuration language: Kotlin DSL (build.gradle.kts) [Recommended]

3. Run your app. The app should look like the screenshot below.


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
Happy Birthday - MainActivity.kt [Happy_Birthday.app.main] - Administrator
HappyBirthday2 › app
Android
app
manifests
kotlin+java
res
res (generated)
Gradle Scripts
activity_main.xml
MainActivity.kt
Running Devices:
Pixel 5 API 31
Hello World!
Build: Sync Build Output Build Analyzer
Version Control Run TODO Problems Terminal App Quality Insights Profiler App Inspection Logcat Services Build

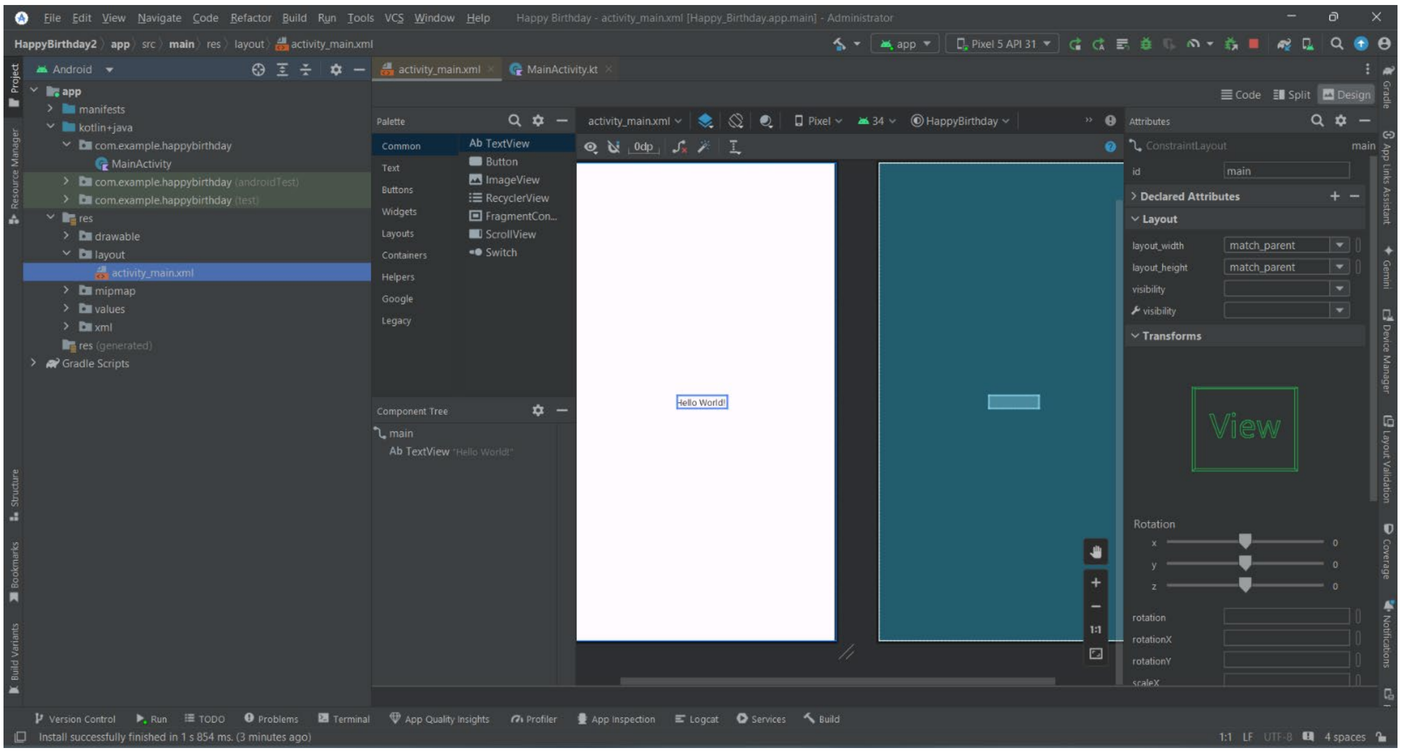
```
package com.example.happybirthday

import ...

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.
            val systemBars = insets.getInsets(WindowInsetsCompat.
            v.setPadding(systemBars.left, systemBars.top, systemB
            insets
        }
    }
}
```

```
Build HappyBirthday2: finished At 10/7/2024 5:33 PM    1 min, 2 sec, 120 ms
  Download info

Deprecated Gradle features were used in this build, making it incompatible with Gradle 9.0.

You can use '--warning-mode all' to show the individual deprecation warnings and determine if they come from your own

For more on this, please refer to https://docs.gradle.org/8.7/userguide/command_line_interface.html#sec:command_line_w

BUILD SUCCESSFUL in 1m 1s
32 actionable tasks: 32 executed

Build Analyzer results available
```

Install successfully finished in 1 s 854 ms. (moments ago)

## Change the Hello World message

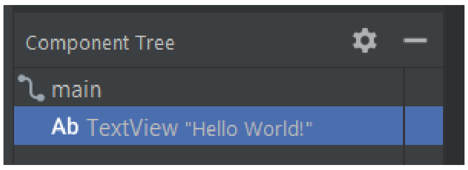
1. In Android Studio find the Project window on the left
2. The activity_main.xml file contains a description of your screen layout.
3. Expand the app folder, then the res folder, and then the layout folder.
4. Double-click on activity_main.xml. This opens activity_main.xml in the Layout Editor and shows the layout it describes in the Design view.



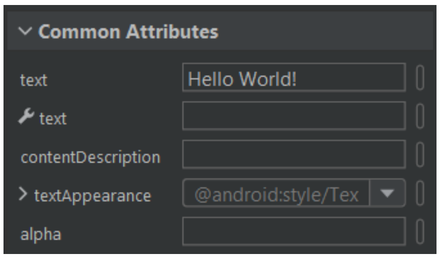
id: main
layout_width: match_parent
layout_height: match_parent

Install successfully finished in 1 s 854 ms. (3 minutes ago)

5. Look at the list of views in the Component Tree. Notice that there is a ConstraintLayout, and a TextView underneath it. These represent the UI of your app. The TextView is indented because it is inside the ConstraintLayout. As you add more Views to the ConstraintLayout, they will be added to this list.
6. Notice that the TextView has "Hello World!" next to it, which is the text you see when you run the app.



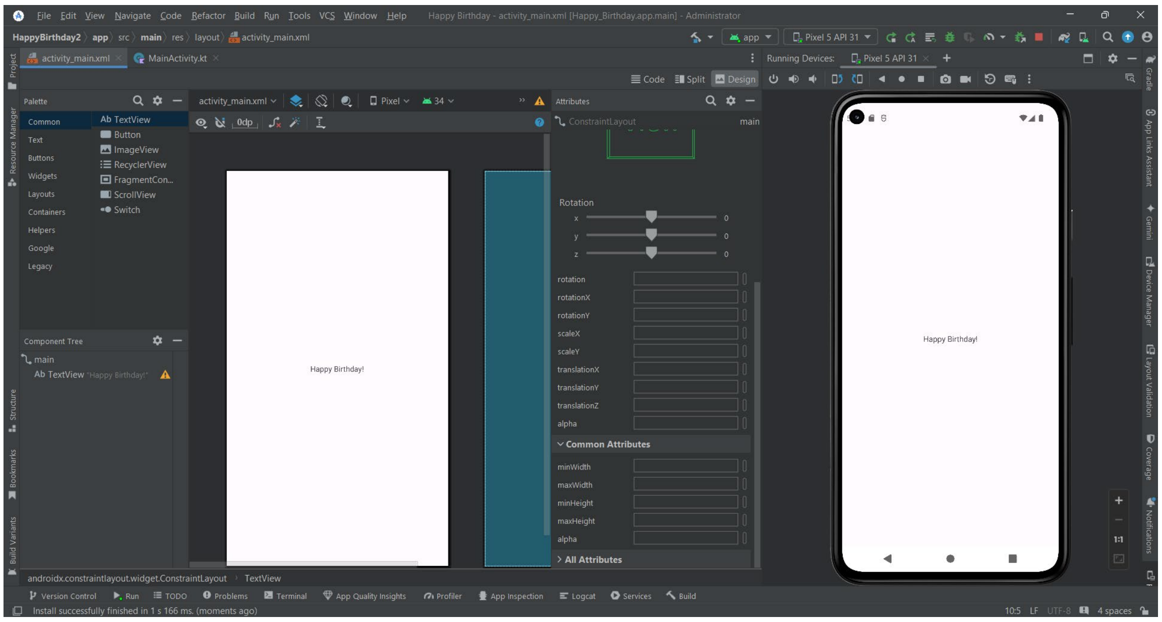
7. In the Component Tree, click on the TextView.
8. Find Attributes on the right.
9. Find the Declared Attributes section.
10. Notice that the text attribute in the Declared Attributes section contains Hello World!



Common Attributes

| | |
|---|---|
| text | Hello World! |
| text | |
| contentDescription | |
| textAppearance | @android:style/Tex |
| alpha | |

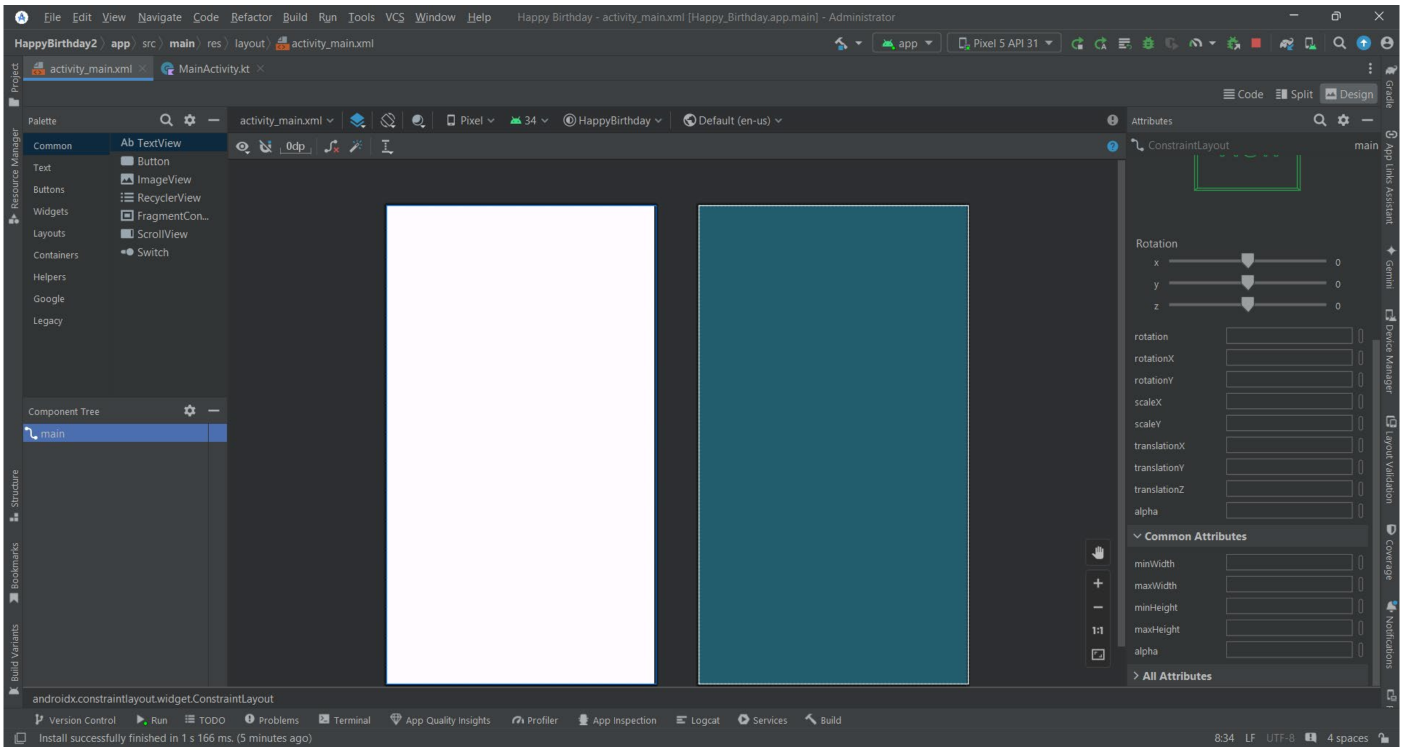
11. Click on the text attribute where the Hello World! text is.
12. Change it to say Happy Birthday!, then press the Enter key.
13. Notice that the text has changed in the Design View.....(this is cool, you can see your changes right away!)
14. Run your app, and now it says Happy Birthday!


Happy Birthday - activity_main.xml [Happy_Birthday.app.main] - Administrator
activity_main.xml
MainActivity.kt
Palette
Component Tree
main
Ab TextView "Happy Birthday!"
Happy Birthday!
Attributes
Common Attributes
All Attributes
Running Devices: Pixel 5 API 31
Happy Birthday!
Install successfully finished in 1 s 166 ms. (moments ago)

## Add TextViews to the layout

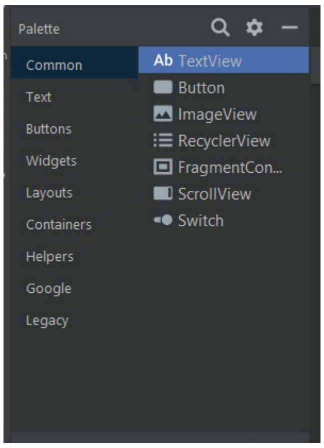
### Delete the current TextView

1. In the Layout Editor, click to select the TextView at the center of the layout
2. Press the Delete key. Android Studio deletes the TextView, and your app now shows just a ConstraintLayout in the Layout Editor and the Component Tree.

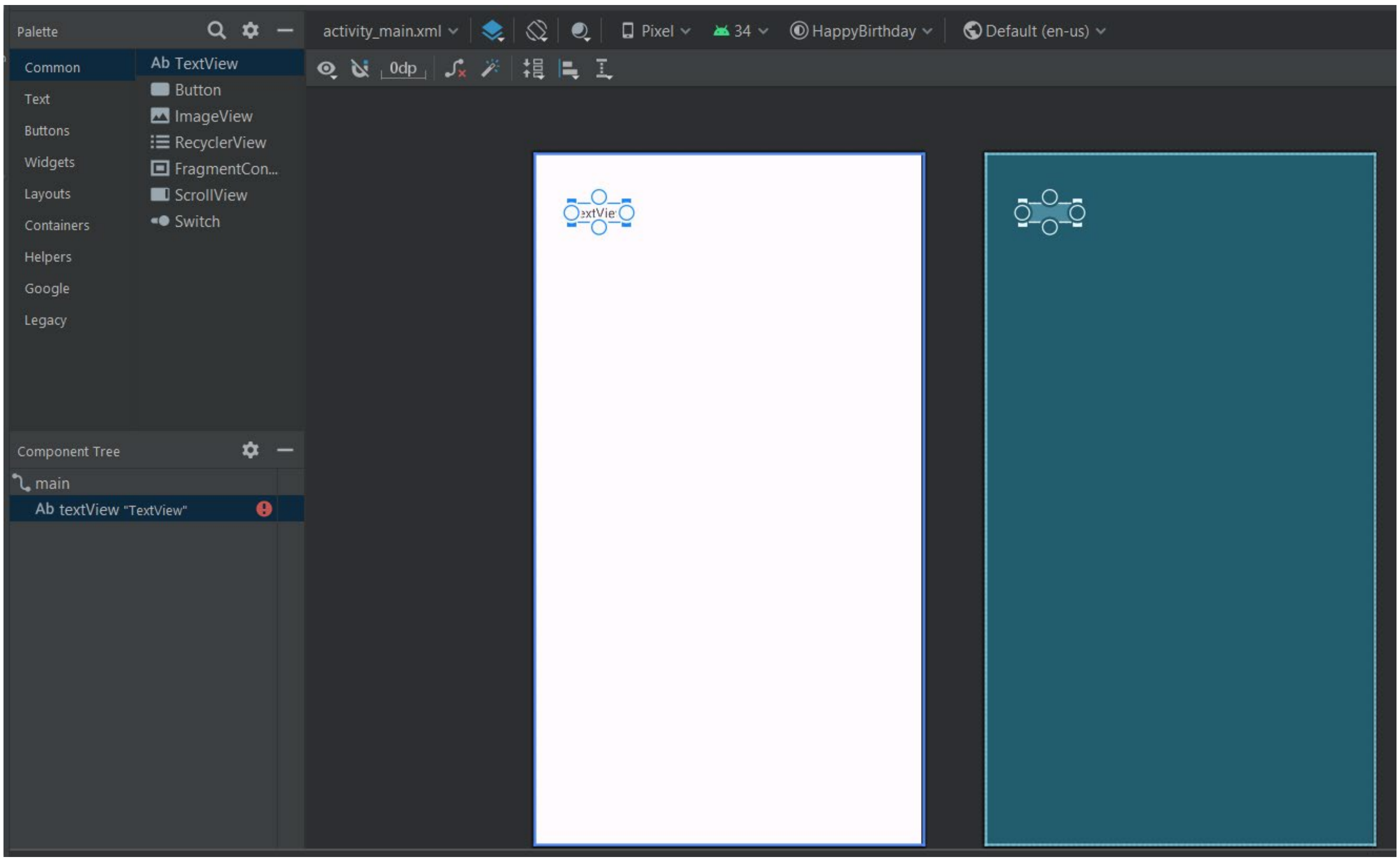


### Add a TextView

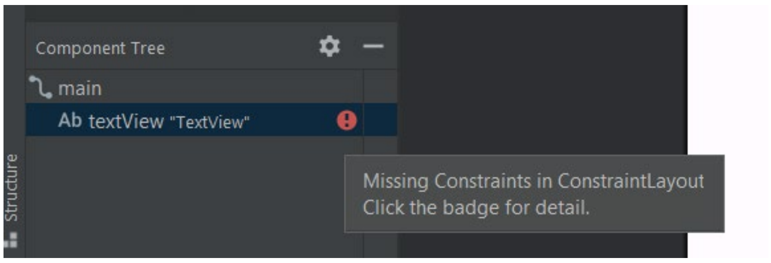
1. Find TextView. It appears both in the Common category and the Text category.



2. Drag a TextView from the Palette to the upper left of the design surface in the Layout Editor and drop it. You don't need to be exact, just drop it near the upper left corner.



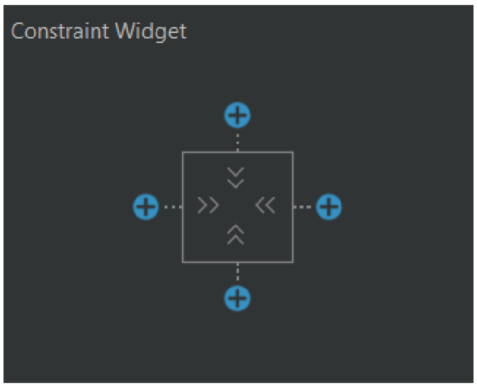
3. Notice that there's a TextView added, and notice a red exclamation mark in the Component Tree.
4. Hover your pointer over the exclamation mark, and you'll see a warning message that the view is not constrained and will jump to a different position when you run the app. You'll fix that in the next step.



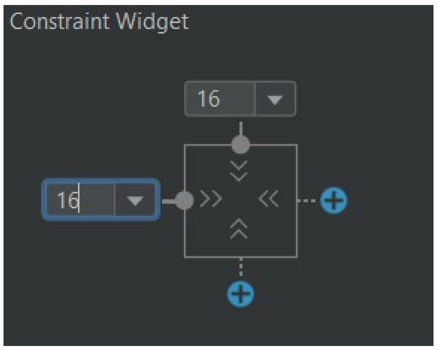
Missing Constraints in ConstraintLayout
Click the badge for detail.

## Position the TextView

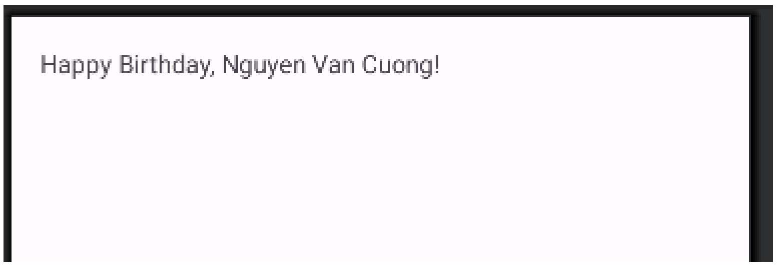
1. In Attributes on the right, find the Constraint Widget in the Layout section. The square represents your view.



2. Click on the + at the top of the square. This is for the constraint between the top of your text view and the top edge of the constraint layout.
3. A field with a number appears for setting the top margin. The margin is the distance from the TextView to the edge of the container, the ConstraintLayout. The number you see will be different depending on where you dropped the TextView. When you set the top margin, Android Studio automatically also adds a constraint from the top of the text view to the top of the ConstraintLayout.
4. Change the top margin to 16.
5. Do the same for the left margin.

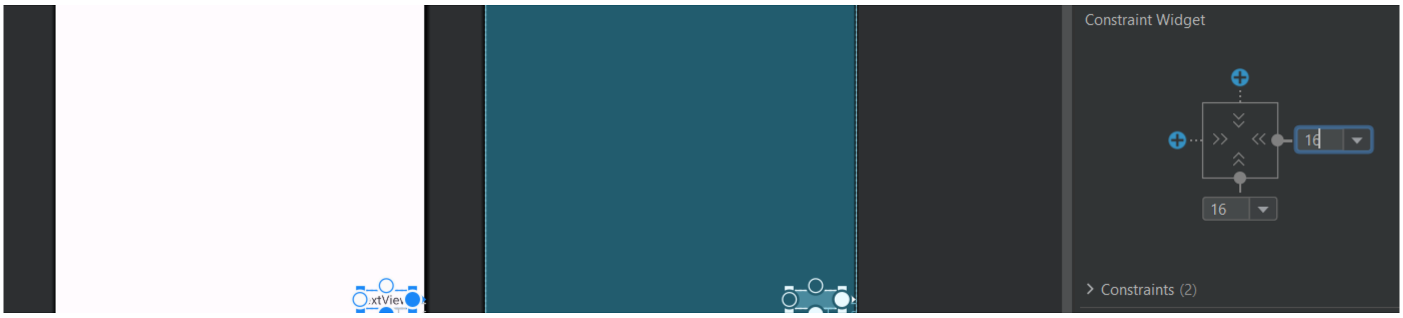


6. Set the text to wish your friend a happy birthday, for example "Happy Birthday, Nguyen Van Cuong!" and press Enter.
7. Notice that the Design view updates to show what your app will look like.

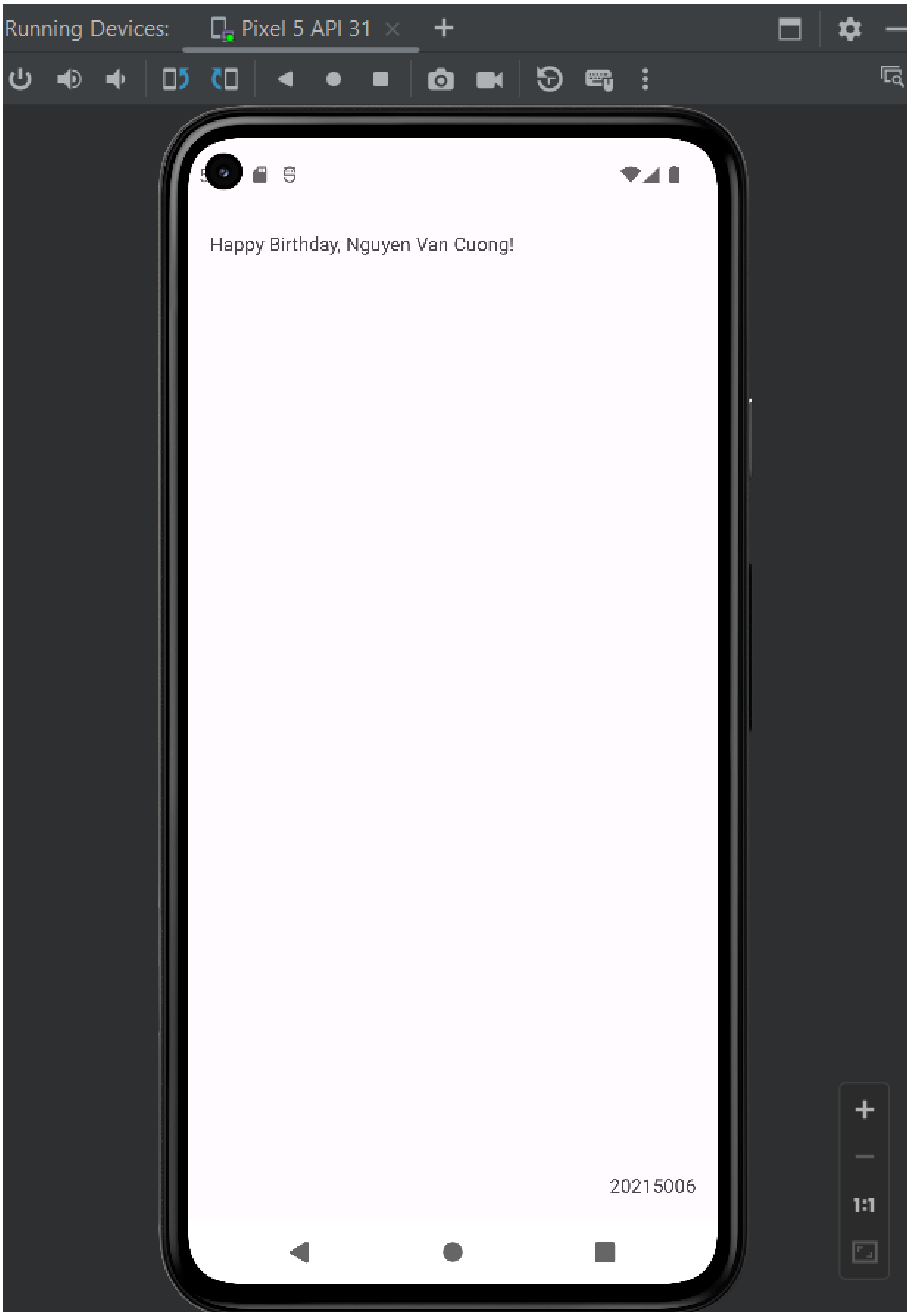

Happy Birthday, Nguyen Van Cuong!

## Add and position another TextView

1. Drag a new TextView from the Palette and drop it near the bottom right of the app view in the Layout Editor.
2. Set a right margin of 16.
3. Set a bottom margin of 16.



4. In Attributes, set the text attribute to sign your card, for example "20215006"
5. Run your app. You should see your birthday wish in the upper left and your signature in the lower right.

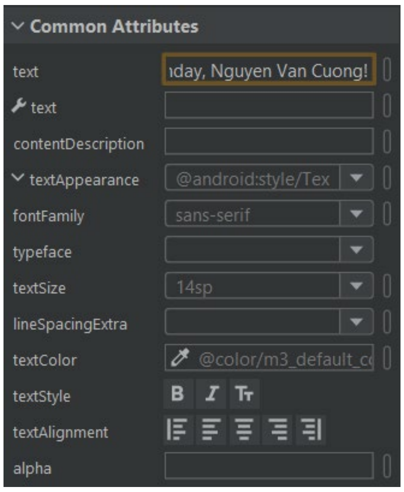


Happy Birthday, Nguyen Van Cuong!

20215006

## Add style to the text

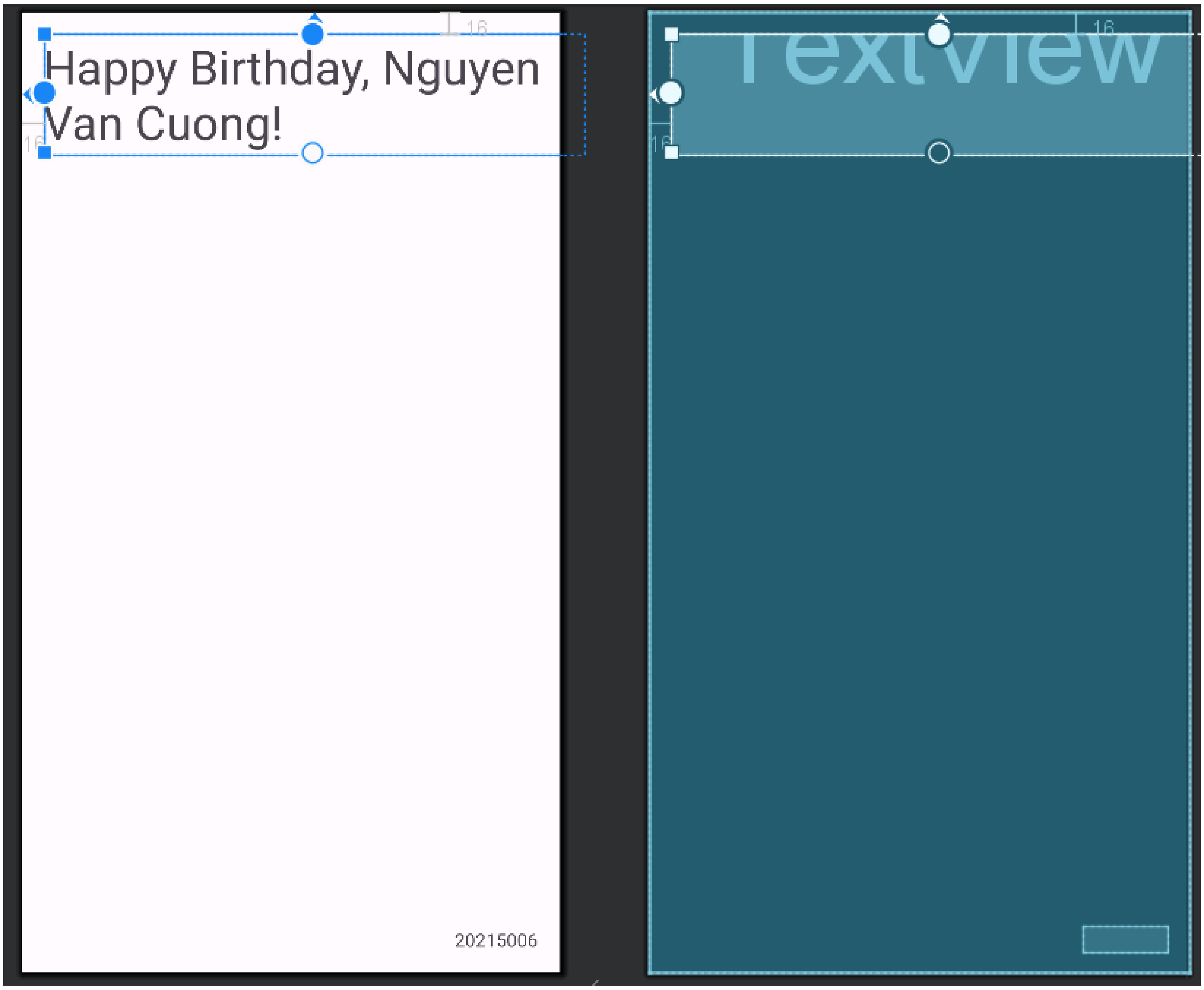
1. Click on the first TextView in the Component Tree and find the Common Attributes section of the Attributes window. You may need to scroll down to find it.
2. Notice the various attributes including fontFamily, textSize, and textColor



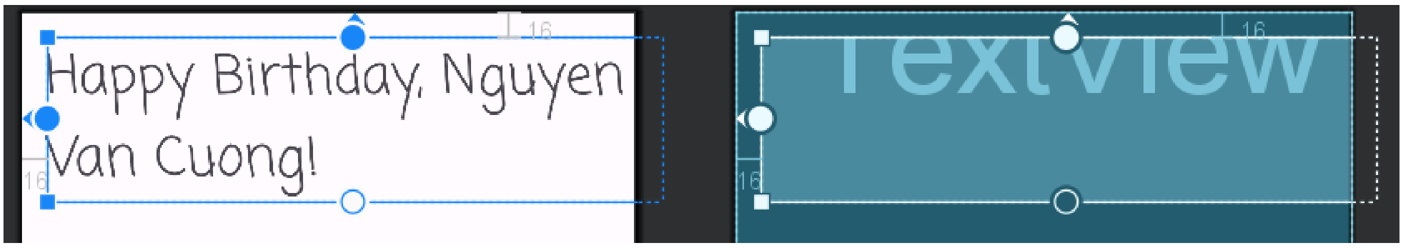
Common Attributes

| | |
|---|---|
| text | ıday, Nguyen Van Cuong! |
| text | |
| contentDescription | |
| textAppearance | @android:style/Tex |
| fontFamily | sans-serif |
| typeface | |
| textSize | 14sp |
| lineSpacingExtra | |
| textColor | @color/m3_default_c |
| textStyle | |
| textAlignment | |
| alpha | |

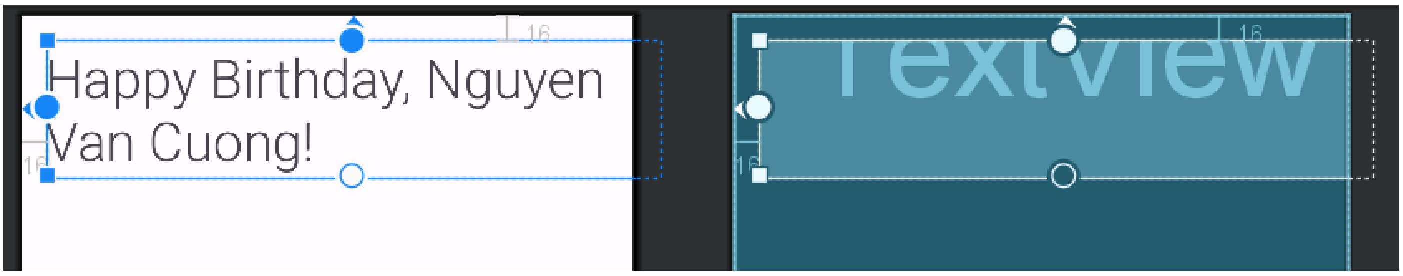
3. Look for textAppearance.
4. If textAppearance is not expanded, click on the down triangle.
5. In the textSize set the textSize to 36sp.
6. Notice the change in the Layout Editor.



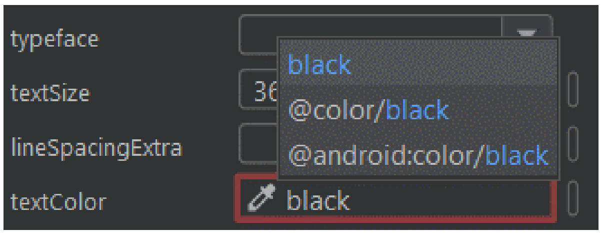
7. Change the fontFamily to casual.



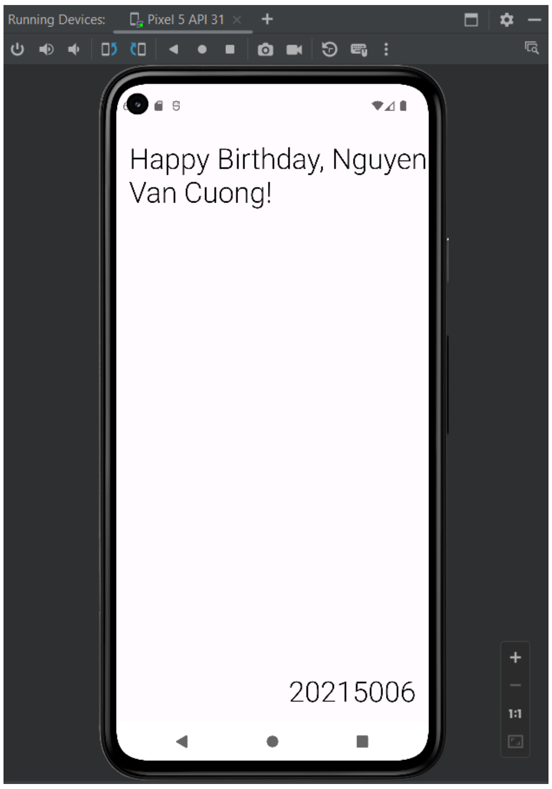
8. Try some different fonts to see what they look like. There are even more choices for fonts at the bottom of the list, under More Fonts...
9. When you're done trying different fonts, set the fontFamily to sans-serif-light.



10. Click on the textColor attribute's edit box and start typing black. Notice that as you type, Android Studio shows a list of colors that contain the text you've typed so far.



11. Select @android:color/black from the list of colors and press Enter.
12. In the TextView with your signature, change the textSize, textColor and fontFamily to match.
13. Run your app and look at the result.

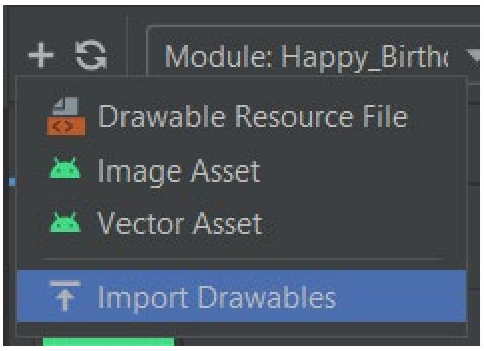


## 4.2. Add images to your Android app

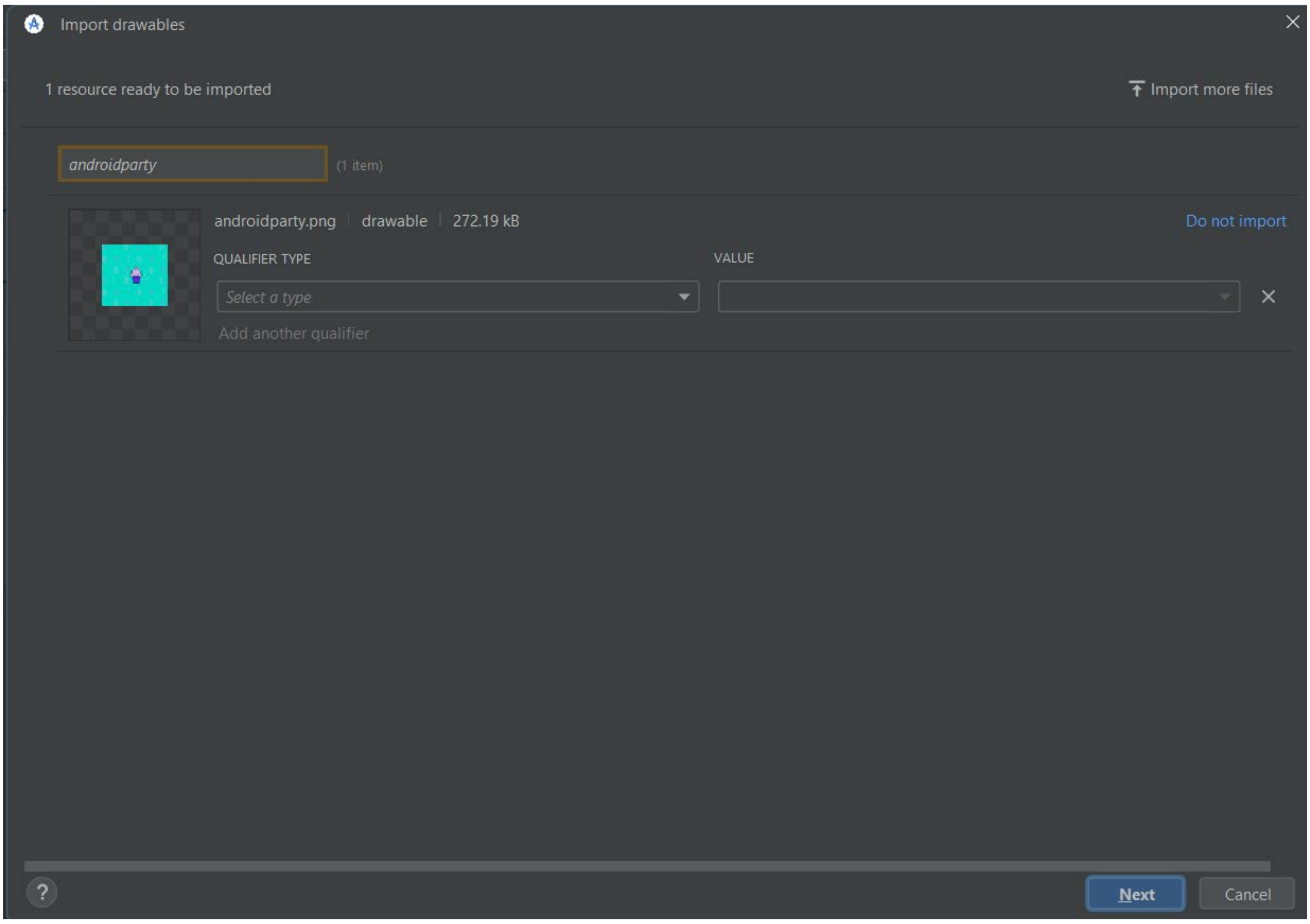
### Set up your app: In lab 4.1

### Add an image to your project

1. Access the image for your birthday card on Github.
2. Click the Download button on the right. This displays the image by itself
3. Right-click on the image and save the file to your computer as androidparty.png. Make note of where you saved it (for example, the Downloads folder).
4. In Android Studio, click on View > Tool Windows > Resource Manager in the menus or click on the Resource Manager tab to the left of the Project window.
5. Click the + below Resource Manager, and select Import Drawables. This opens a file browser.



6. In the file browser find the image file you downloaded and click Open.



1 resource ready to be imported

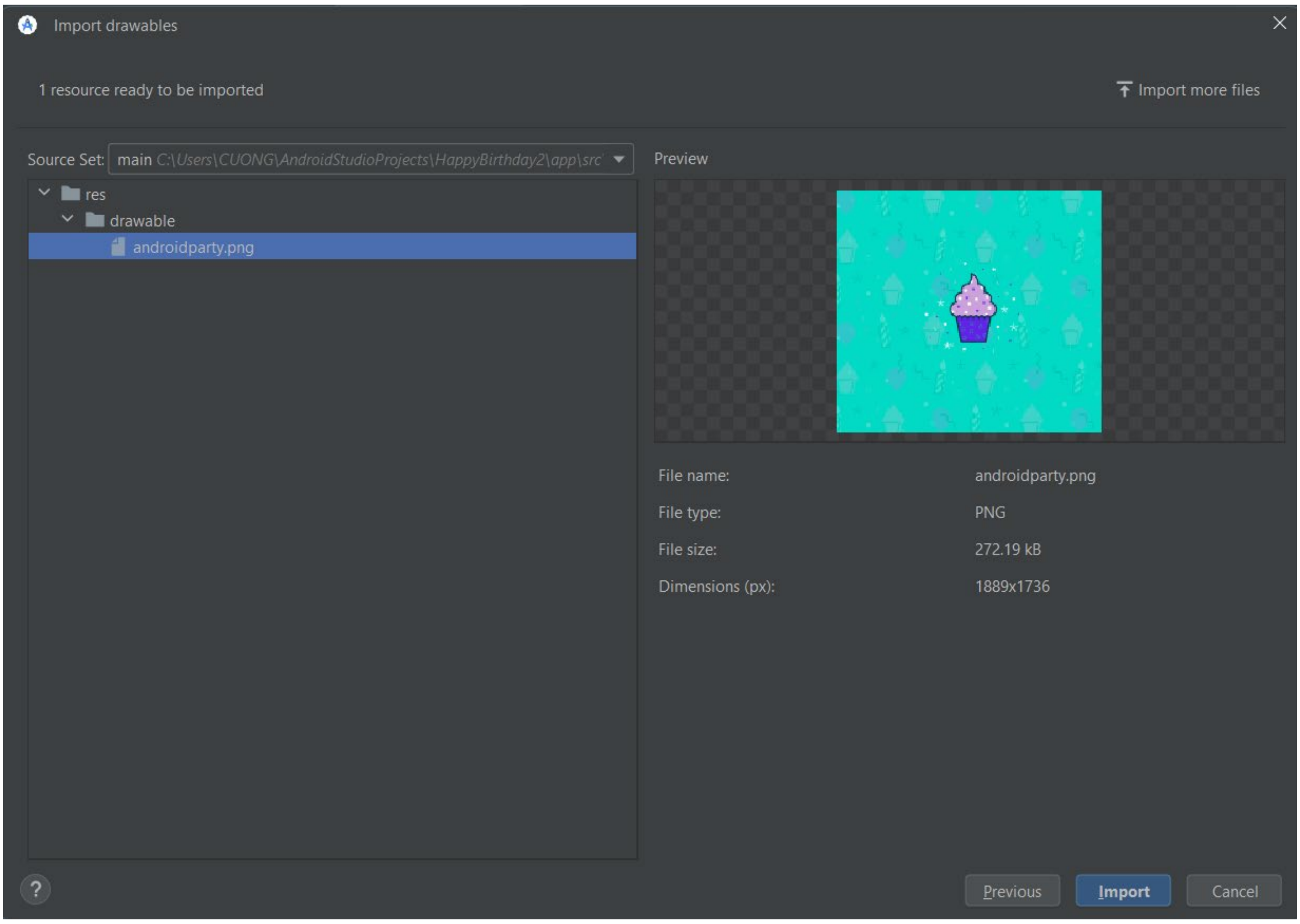
androidparty (1 item)

androidparty.png | drawable | 272.19 kB

QUALIFIER TYPE: Select a type

VALUE:

7. Click Next. Android Studio shows you a preview of the image.


Import drawables

1 resource ready to be imported

Import more files

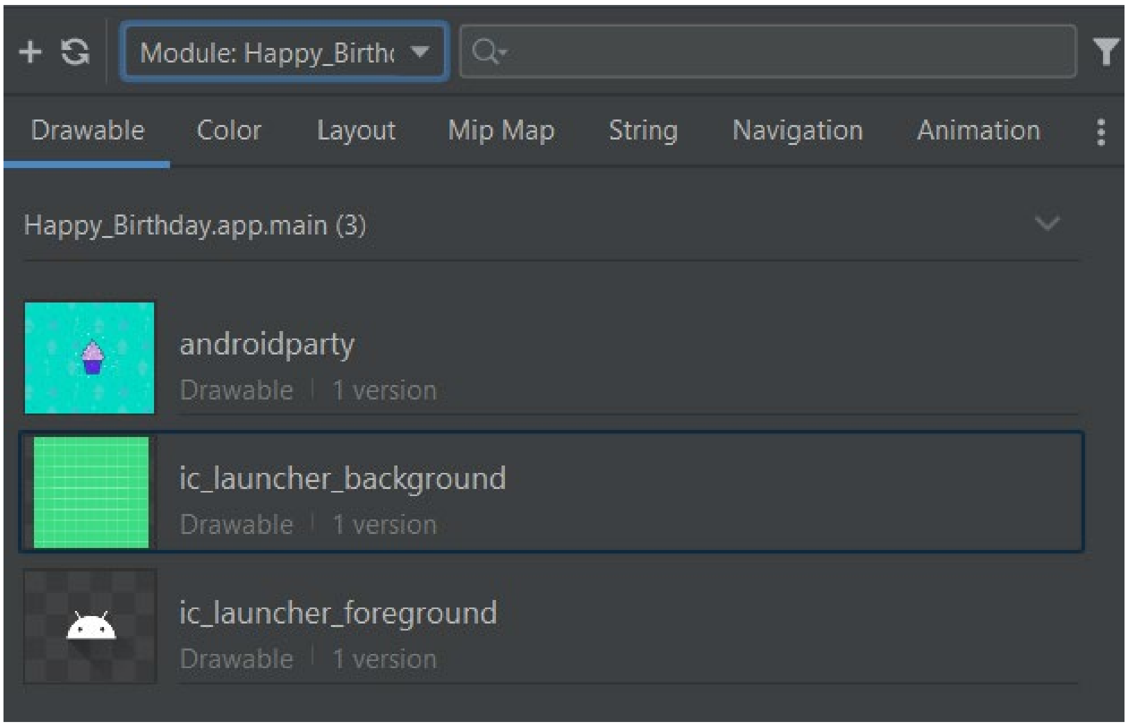
Source Set: main C:\Users\CUONG\AndroidStudioProjects\HappyBirthday2\app\src

res
drawable
androidparty.png

Preview

File name: androidparty.png
File type: PNG
File size: 272.19 kB
Dimensions (px): 1889x1736

8. Click Import.
9. If the image has been imported successfully, Android Studio adds the image to your Drawable list. This list includes all your images and icons for the app. You can now use this image in your app.



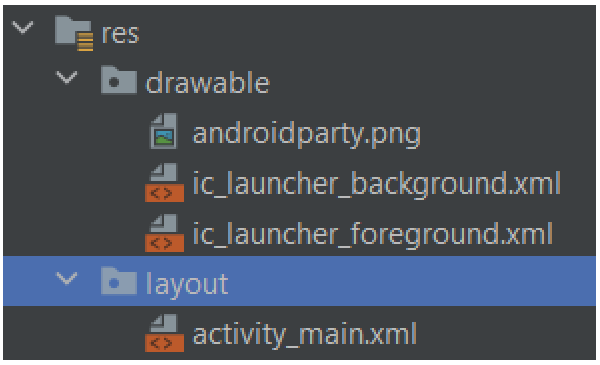
Happy_Birthday.app.main (3)

androidparty
Drawable | 1 version

ic_launcher_background
Drawable | 1 version

ic_launcher_foreground
Drawable | 1 version

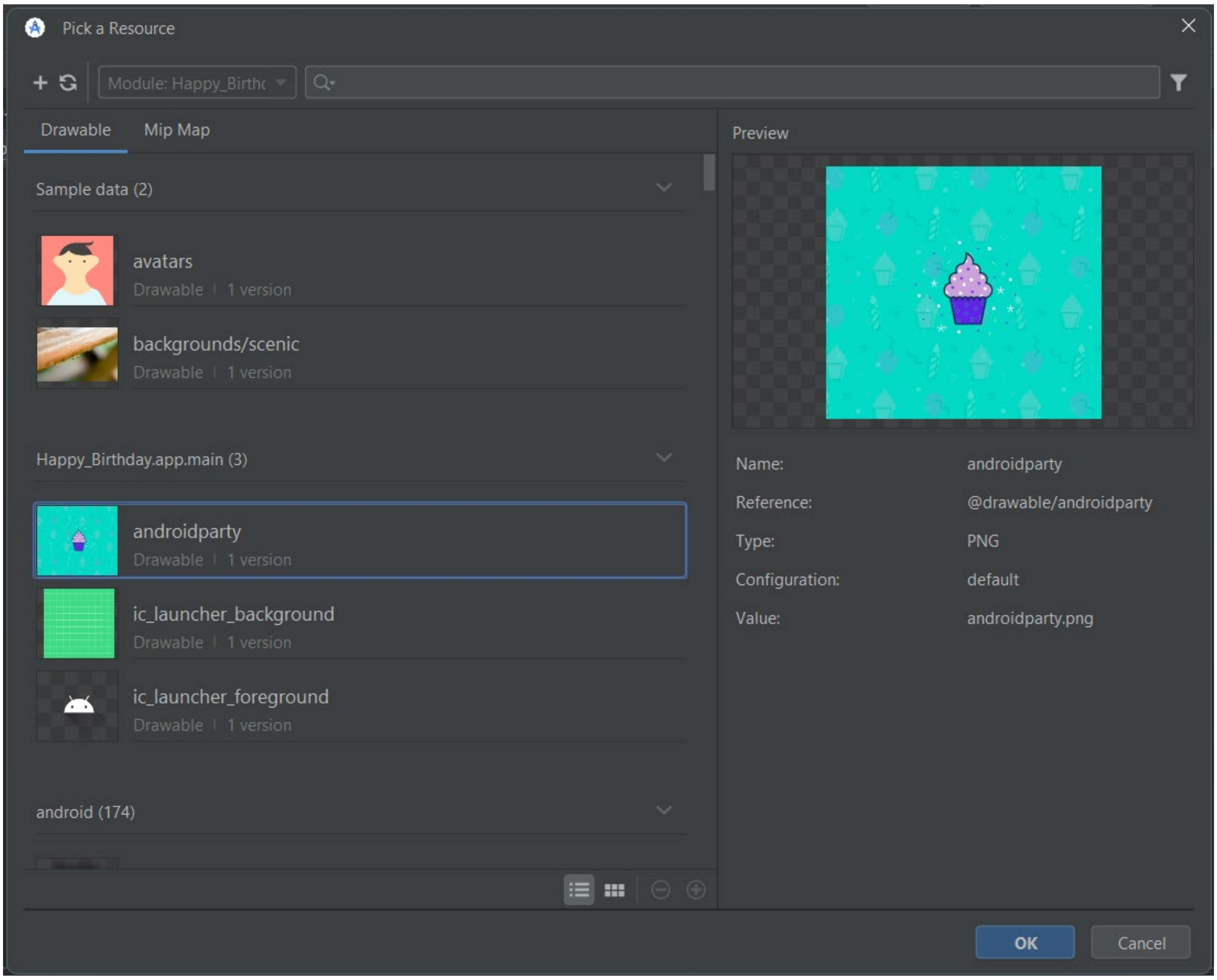
10. Switch back to the project view by clicking on View > Tool Windows > Project in the menus or the Project tab on the far left.
11. Confirm the image is in the drawable folder of your app by expanding app > res > drawable.



## Add an ImageView

### Add an ImageView and set its image

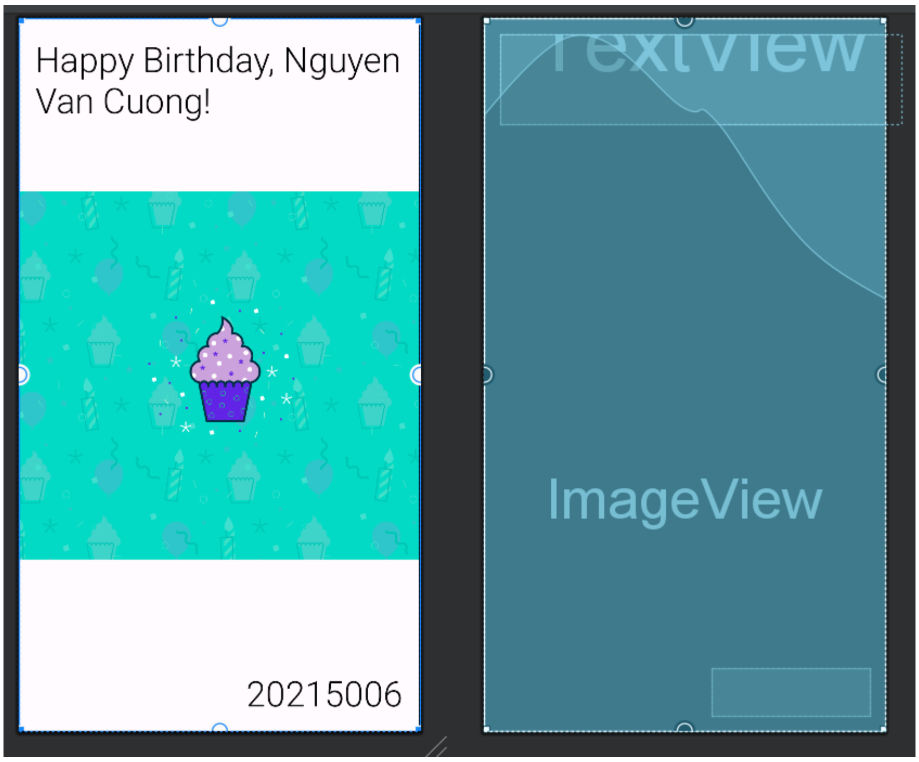
1. In the Project window open activity_main.xml ( app > res > layout > activity_main.xml ).
2. In the Layout Editor, go to the Palette and drag an ImageView to your app. Drop it near the center and not overlapping any of the text
3. In the Pick a Resource dialog find the cake image in the Drawable list.
4. Click on the image and then click OK.



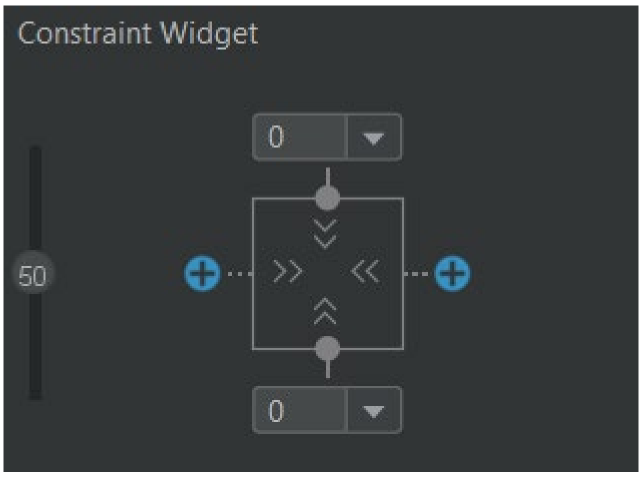
Name: androidparty
Reference: @drawable/androidparty
Type: PNG
Configuration: default
Value: androidparty.png

## Position and size the ImageView

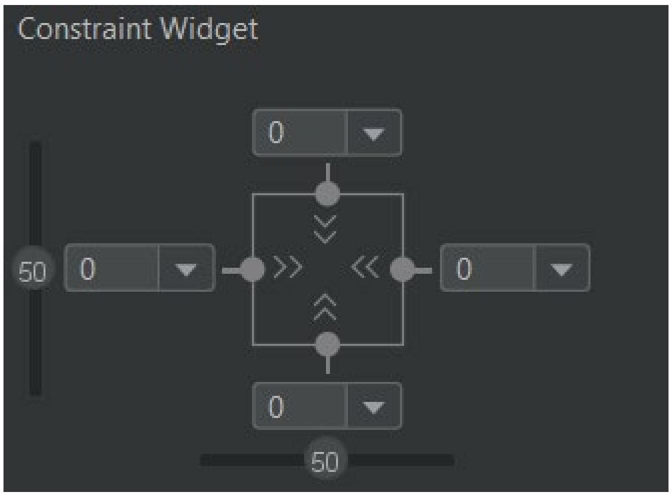
1. Click and drag the ImageView around in the Layout Editor, and note that as you drag, a pink rectangle appears around the app screen in the Design view. The pink rectangle indicates the boundaries of the screen for positioning your ImageView.
2. Drop the ImageView so that the left and right edges align with the pink rectangle. Android Studio will "snap" the image to the edges when you're close. (You'll take care of the top and bottom in a moment.)
3. Hold the pointer over the circle at the top of the outline of the ImageView, and it highlights with another circle.
4. Drag the circle towards the top of the app screen, and an arrow connects the circle to your pointer as you are dragging. Drag until it snaps to the top of the screen. You've added a constraint from the top of the ImageView to the top of the ConstraintLayout.



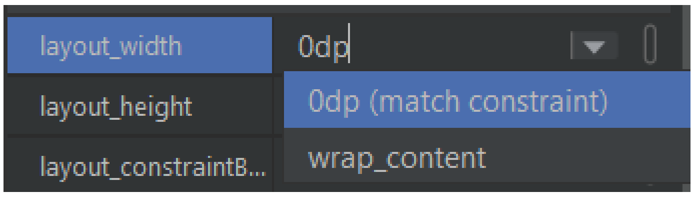
5. Add a constraint from the bottom of the ImageView to the bottom of the ConstraintLayout. It may be too close to the edge to drag as you did for the top. In that case, you can click on the bottom + in the Constraint Widget in the Attributes window to add a constraint. Make sure the margin is 0.



6. In the Attributes pane, use the Constraint Widget to add a margin of 0 to the left and to the right sides. This automatically creates a constraint in that direction.

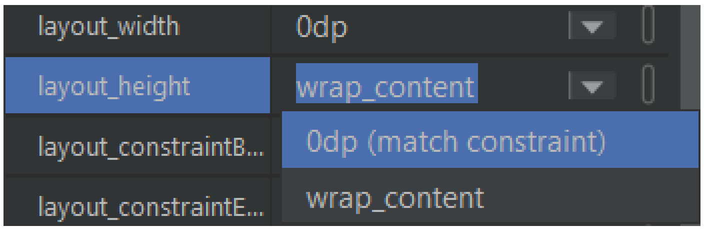


7. Below the Constraint Widget in the Constraints section, set the layout_width to 0dp (match constraint). 0dp is a shorthand to tell Android Studio to use match constraint for the width of the ImageView. Because of constraints you just added, this makes it as wide as the ConstraintLayout, minus any margins.



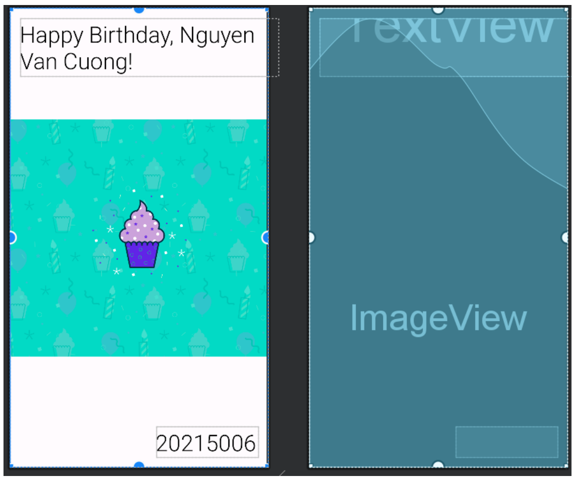
layout_width: 0dp
layout_height
layout_constraintB...

8. Set the layout_height to 0dp (match constraint). Because of the constraints you added, this makes the ImageView as tall as the ConstraintLayout, minus any margins.

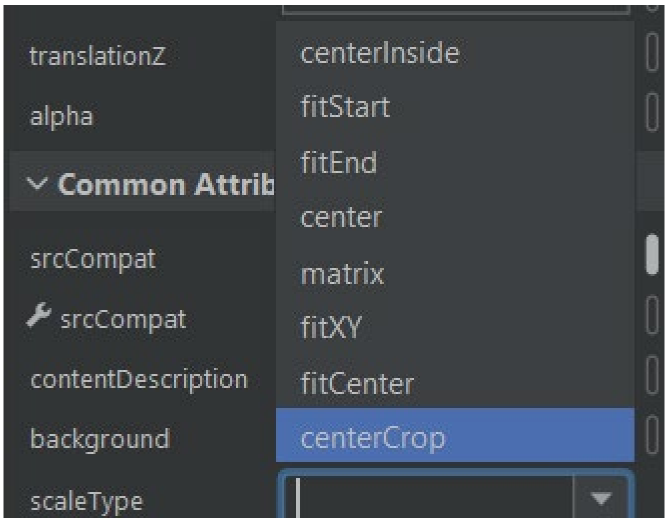


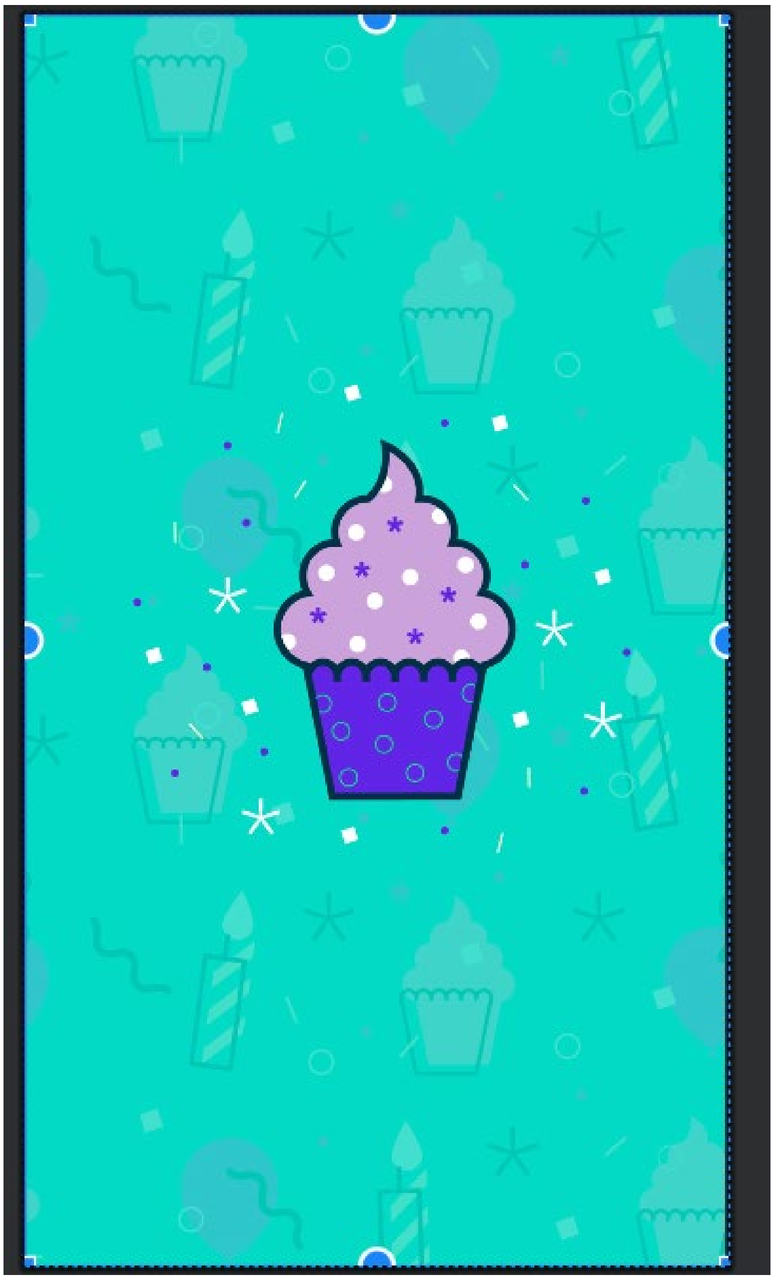
layout_width: 0dp
layout_height: wrap_content
layout_constraintB...
layout_constraintE...

9. The ImageView is as wide and tall as the app screen, but the image is centered inside the ImageView and there is a bunch of whitespace above and below the image. Your app should now look as shown below.



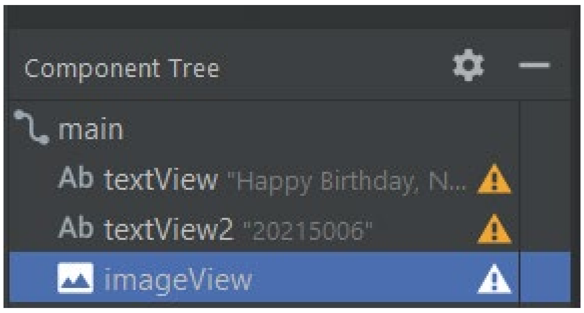
10. Find the scaleType attribute. You may need to scroll down or search for this attribute. Try setting different values for the scaleType to see what they do.
11. When you're done, set the scaleType to centerCrop because that makes the image fill the screen without distorting it.

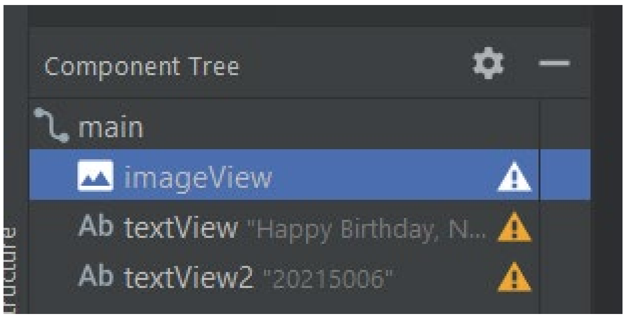


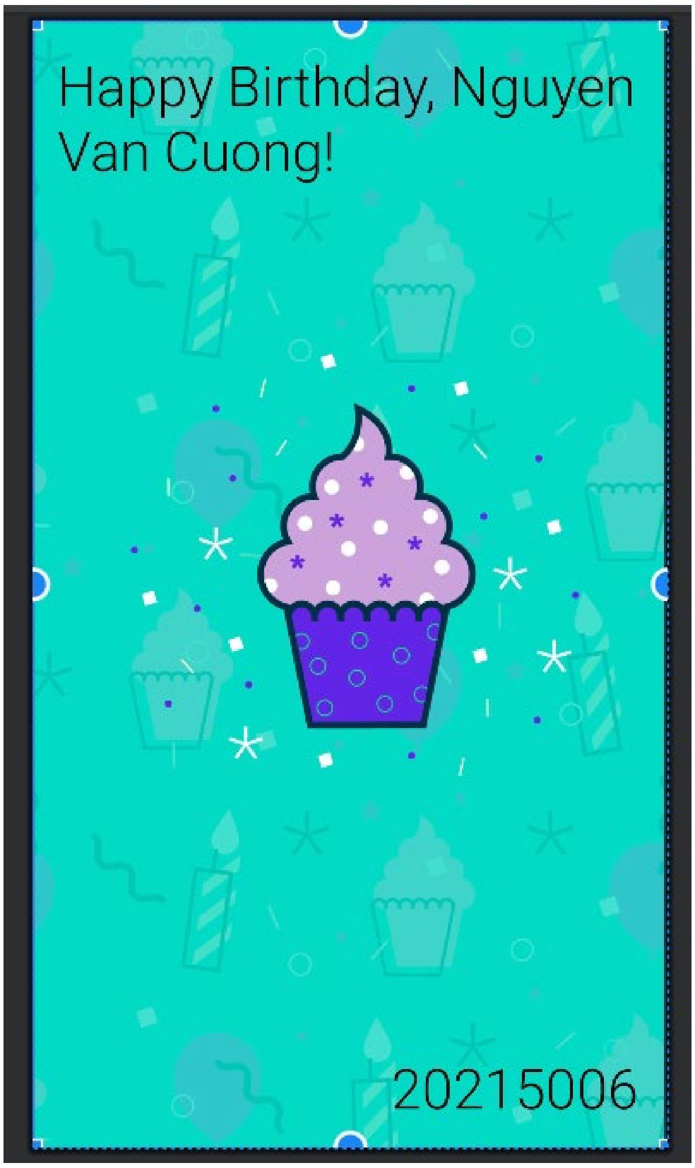


## Move the ImageView behind the text

In the Component Tree, click on the ImageView, and drag it above the TextViews to just below the ConstraintLayout. A blue line with a triangle appears showing where the ImageView will go. Drop the ImageView just below the ConstraintLayout



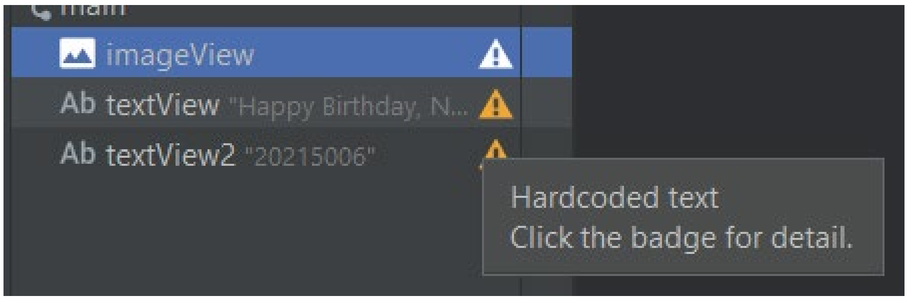




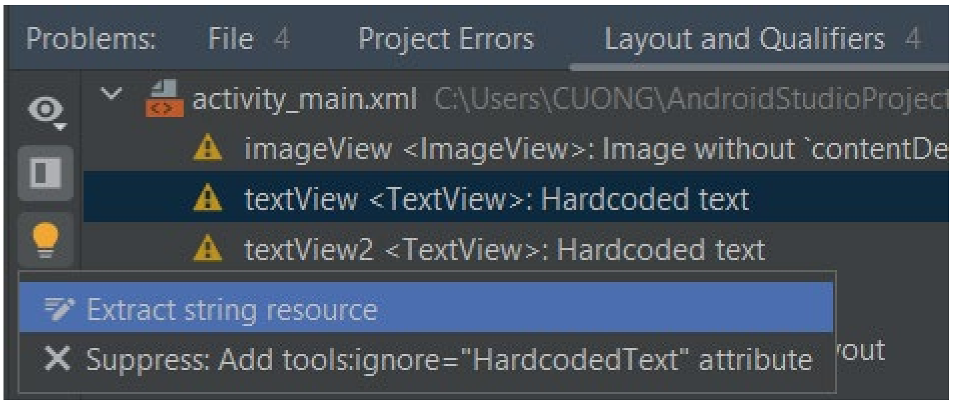
## Adopt good coding practices

### Translating

1. Click on the orange triangle next to the first TextView with "Happy Birthday, Sam!" Android Studio opens a window with more information and a suggested fix. You may need to scroll down to see the Suggested Fix.

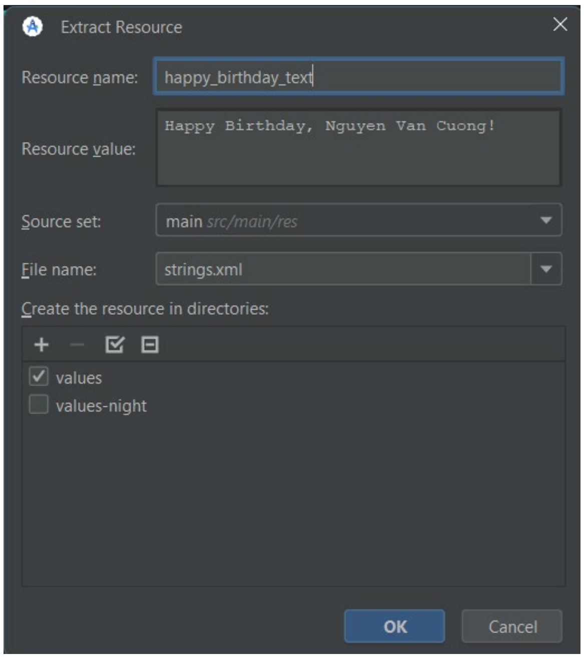


Hardcoded text
Click the badge for detail.


Problems: File 4 Project Errors Layout and Qualifiers 4
Extract string resource
Suppress: Add tools:ignore="HardcodedText" attribute

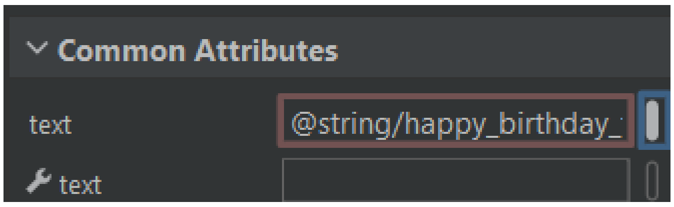
activity_main.xml C:\Users\CUONG\AndroidStudioProject
imageView <ImageView>: Image without `contentDe
textView <TextView>: Hardcoded text
textView2 <TextView>: Hardcoded text

2. Click the Fix button. Android Studio opens the Extract Resource dialog. In this dialog, you can customize what your string resource is called, and some details on how to store it. The Resource name is what the string is going to be called. The Resource value will be the actual string itself.
3. In the Extract Resource dialog, change the Resource name to happy_birthday_text. String resources should have lowercase names, and multiple words should be separate with an underscore. Leave the other settings with the defaults.



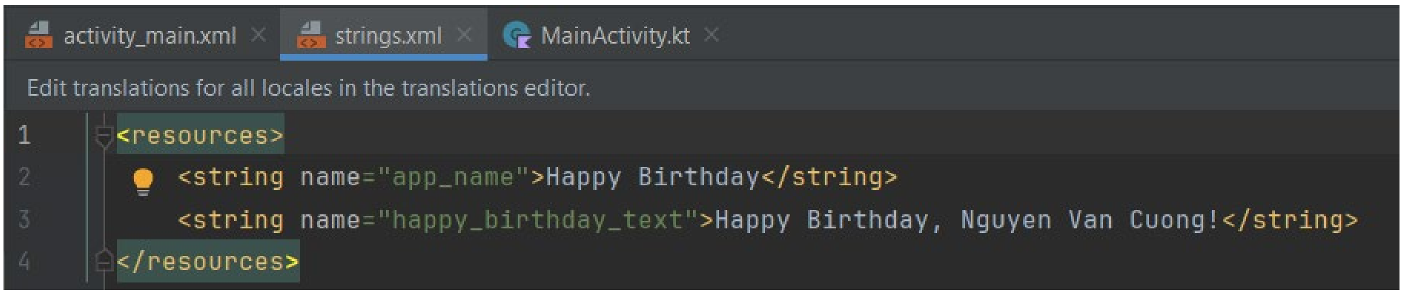
Resource name: happy_birthday_text
Resource value: Happy Birthday, Nguyen Van Cuong!
Source set: main src/main/res
File name: strings.xml
Create the resource in directories:
✓ values
values-night

4. Click the OK button.
5. In the Attributes pane, find the text attribute, and notice that Android Studio has automatically set it to @string/happy_birthday_text for you.



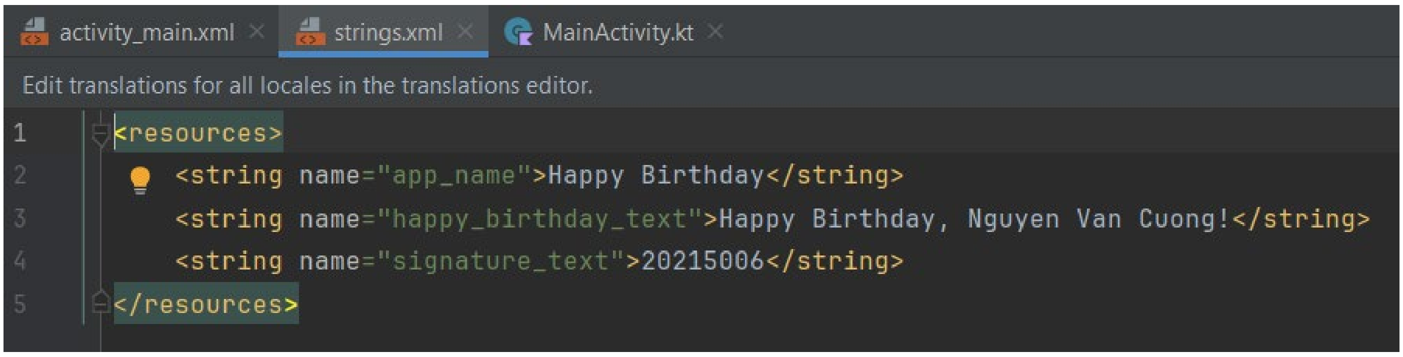
text: @string/happy_birthday_

6. Open strings.xml (app > res > values > strings.xml) and notice that Android Studio has created a string resource called happy_birthday_text.


activity_main.xml
strings.xml
MainActivity.kt
Edit translations for all locales in the translations editor.

```xml
<resources>
    <string name="app_name">Happy Birthday</string>
    <string name="happy_birthday_text">Happy Birthday, Nguyen Van Cuong!</string>
</resources>
```

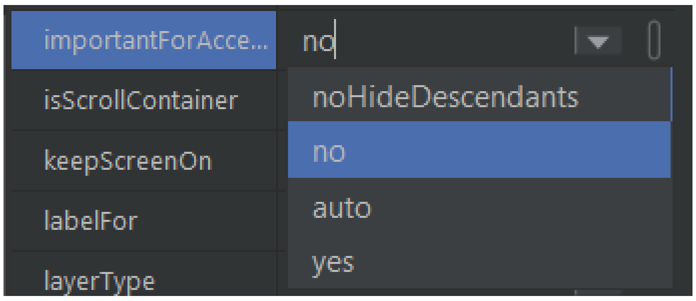
7. Follow the same steps to extract the text for the signature TextView, and name the string resource signature_text.


activity_main.xml
strings.xml
MainActivity.kt
Edit translations for all locales in the translations editor.

```xml
<resources>
    <string name="app_name">Happy Birthday</string>
    <string name="happy_birthday_text">Happy Birthday, Nguyen Van Cuong!</string>
    <string name="signature_text">20215006</string>
</resources>
```

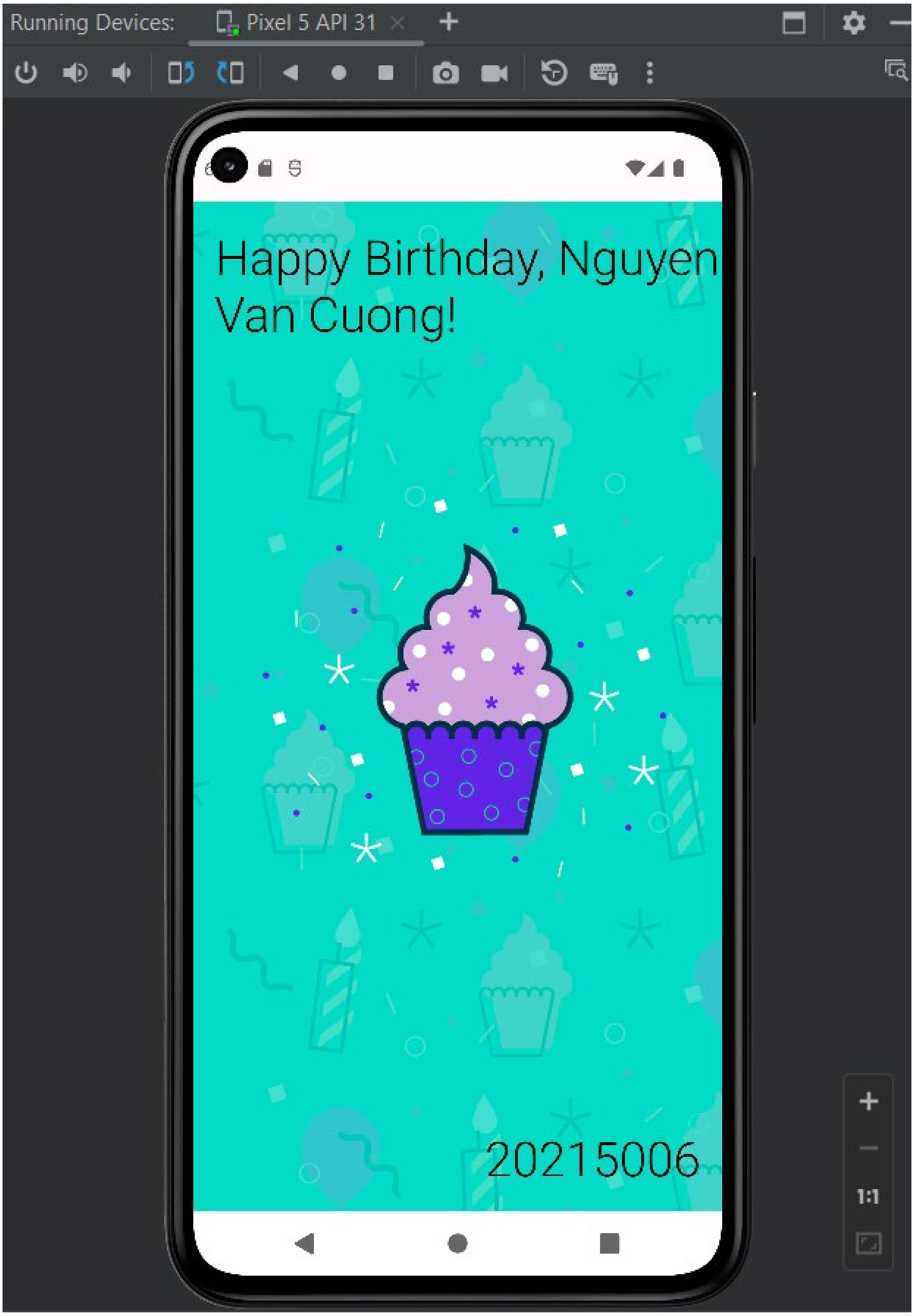
## Check your app for accessibility

1. In the Component Tree, notice the orange triangle next to the ImageView that you added earlier. If you hover the pointer over it, you'll see a tooltip with a warning about a missing 'contentDescription' attribute on the image. A content description can help make your app more usable with TalkBack by defining the purpose of the UI element.
2. In the Component Tree, select the ImageView.
3. In the Attributes window, in the All Attributes section, find importantForAccessibility and set it to no.



importantForAcce...: no

4. Run your app again to make sure it still works.



Happy Birthday, Nguyen Van Cuong!

20215006