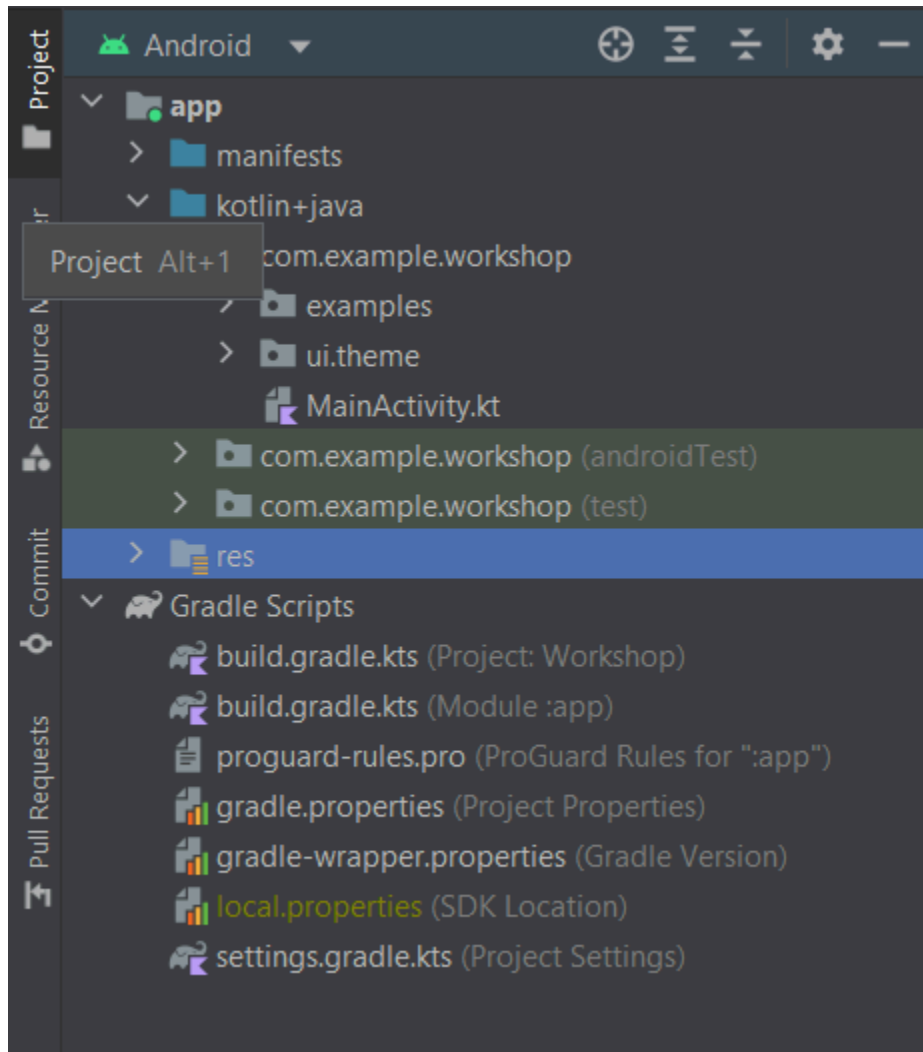
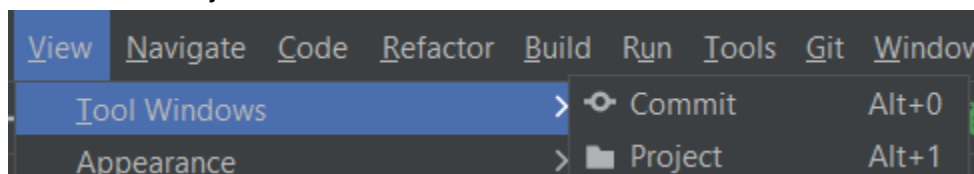


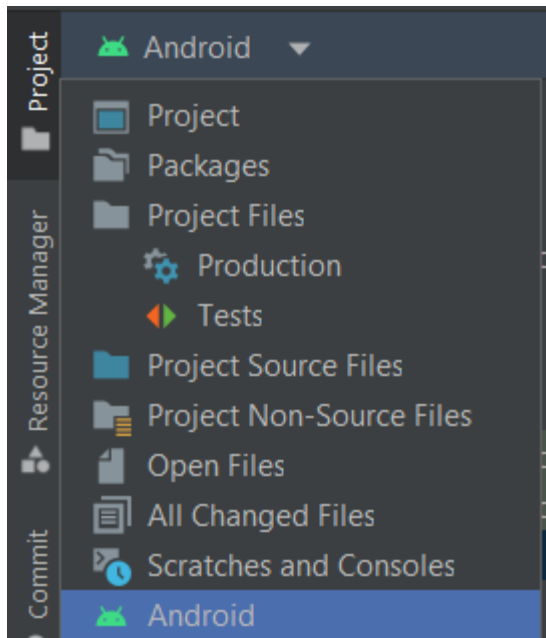
Phase 1: View Project Structure



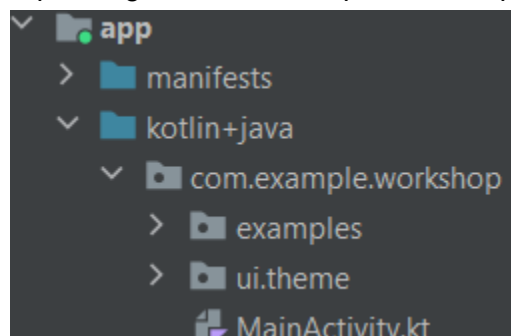
Step 1: Make sure Project View is open using the drop-down menu:: View>>Tool Windows>> Project. Alt+1 shortcut on Windows.



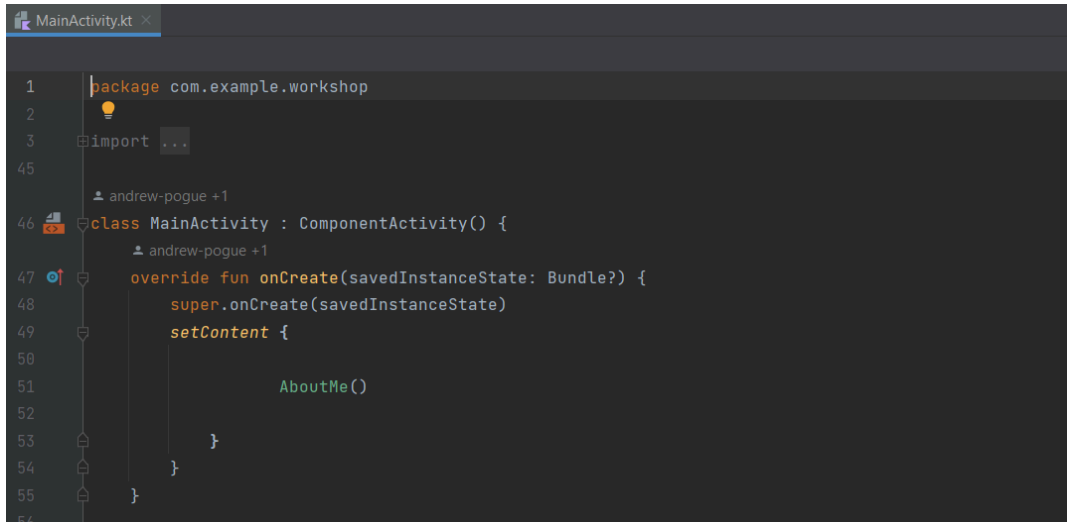
Step 2: Make sure you are using Android format in Project View by using the drop-down menu: Android



Step 3: Expand folders to view MainActivity.kt by expanding the “app” folder and then expanding the “com.example.workshop” package.

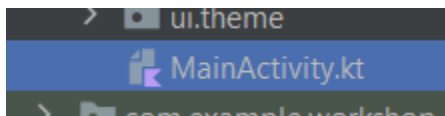


Phase 2: Review the MainActivity.kt example.

A screenshot of an IDE showing the MainActivity.kt file. The code is in Kotlin and defines a class MainActivity that inherits from ComponentActivity. It overrides the onCreate method, which calls super.onCreate, sets the content to AboutMe(), and calls AboutMe().

```
1 package com.example.workshop
2
3 import ...
4
45
46 class MainActivity : ComponentActivity() {
47     override fun onCreate(savedInstanceState: Bundle?) {
48         super.onCreate(savedInstanceState)
49         setContent {
50             AboutMe()
51         }
52     }
53 }
54
55 }
```

Step 1: Open the MainActivity.kt file from the Project View.



Step 2: Identify the different Functions listed below;

The **onCreate** function is like the main function in c or Java, it defines what code is run when the program starts. There are also variations like **onResume** that are used in different situations.

The **AboutMe** function structures the code to be scrollable and to fill out the screen. It also has the other Composables in the order we need. You don't need to edit this function.

The **Title** function is the name of your profile. We will use your name with your customized text.

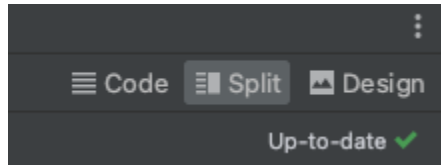
The **ProfilePic** function will be used to share a picture of yourself. We will explore adding resources to the project and connecting them to your code.

The **SelfDescription** function will be used to talk about yourself. We will explore text style and Modifiers to customize your text.

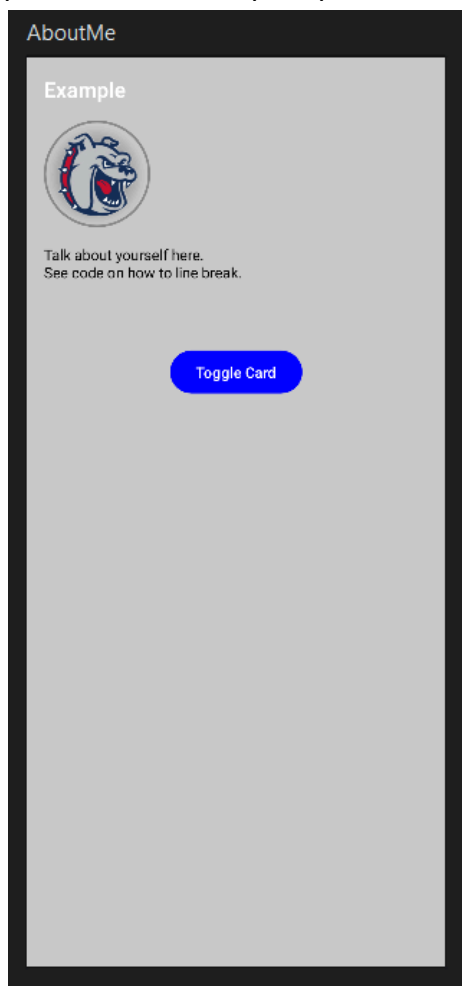
The **YourButton** function is a customized button users can click. We will only modify the button itself and not the other content around it. The button has an “onClick” that will do something when we program it.

The **YourCard** function is filled with content that needs to be rendered. We will edit the content and use buttons to open and close the card. We only modify the text and button in the card to your style.

Step 3: Make sure the split view is selected to see the preview.



Step 4: Make sure your code compiles and you can view the preview. If you don't see the preview, follow the prompt in the middle of the preview window.



Phase 3: Edit the Title of your profile

```

@Composable
fun Title() {

    Text(
        text = "Example",
        style = TextStyle(
            //Choose your own style
            color = Color.White, // Text color
            fontSize = 20.sp, // Text size
            fontWeight = FontWeight.SemiBold, // Font weight
            //textAlign = TextAlign.Center // Text alignment

```

Step 1: In the **Title** function replace the “Example” text with your name.

Step 2: Change the color to your favorite color.

Step 3: Change the font size so your name fills the width of the screen.

Step 4: Change the font size so your name fills half the width of the screen.

Step 5: Remove the // commit and set your text to the center of the screen.

Phase 4: Edit your profile

Self-Description

```

@Composable
fun SelfDescription() {

    Text(
        text = "I am Here and Now." + "\n" + "See code on how to line break.",

        style = TextStyle(
            //Add your own modifiers

```

Step 1: In the **SelfDescription** function replace the Text string with a description about yourself.

Step 2: Select the TextStyle function and then hover over it. This should bring up a list of the function parameters.

```
style = TextStyle(  
  //Ad  
)  
  public constructor TextStyle(  
    color: Color = COMPILED_CODE,  
    fontSize: TextUnit = COMPILED_CODE,  
    fontWeight: FontWeight? = COMPILED_CODE,  
    fontStyle: FontStyle? = COMPILED_CODE,  
    fontSynthesis: FontSynthesis? = COMPILED_CODE,  
    fontFamily: FontFamily? = COMPILED_CODE,  
    fontFeatureSettings: String? = COMPILED_CODE,  
    letterSpacing: TextUnit = COMPILED_CODE,  
    baselineShift: BaselineShift? = COMPILED_CODE,  
    textGeometricTransform: TextGeometricTransform? = COMPILED_CODE,  
    localeList: LocaleList? = COMPILED_CODE,  
    background: Color = COMPILED_CODE,  
    textDecoration: TextDecoration? = COMPILED_CODE,  
    shadow: Shadow? = COMPILED_CODE,  
    drawStyle: DrawStyle? = COMPILED_CODE,  
    textAlign: TextAlign? = COMPILED_CODE,  
    textDirection: TextDirection? = COMPILED_CODE,  
    lineHeight: TextUnit = COMPILED_CODE,  
    textIndent: TextIndent? = COMPILED_CODE,  
    platformStyle: PlatformTextStyle? = COMPILED_CODE,  
    lineHeightStyle: LineHeightStyle? = COMPILED_CODE,  
    lineBreak: LineBreak? = COMPILED_CODE,  
    hyphens: Hyphens? = COMPILED_CODE,  
    textMotion: TextMotion? = COMPILED_CODE  
  )  
)
```

Step 3: Set the text color to something you like.

Step 4: Set the background color to work with your text color.

Step 5: Set the text font size to be readable.

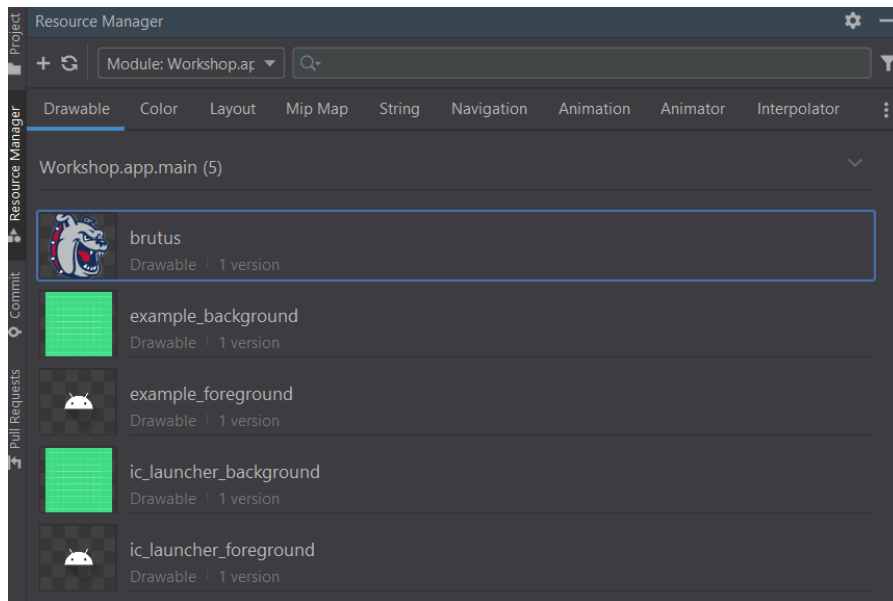
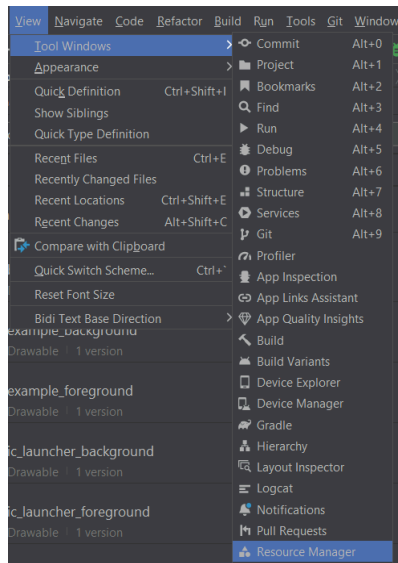
Step 6: Change the font weight to be *Italic*.

Step 7: Set the font alignment.

Step Challenge: Change the font direction.

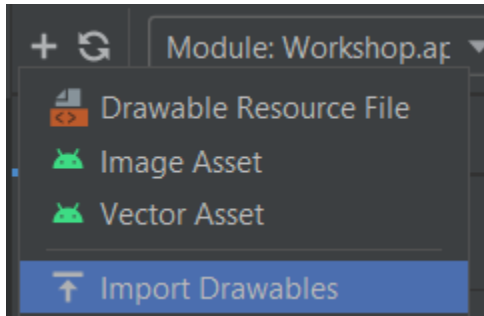
Phase 5: Edit Your Profile Picture

Step 1: Make sure the Resource Manager is open by using the drop-down menu: View>> Tool Windows>> Resource Manager



Step 2: Download a photo of yourself. If you don't want to use your picture, download an animal that looks like you.

Step 3: Add your photo to the project by using, in the Resource Manager, the + button drop-down menu: +>>Import Drawables. Find your picture in your file system, select it, and hit OK.



Step 4: Replace the paintedResource id with your photo resource id. Right-click and copy the id from the Resource Manager or drag the icon from the Resource Manager into your code.

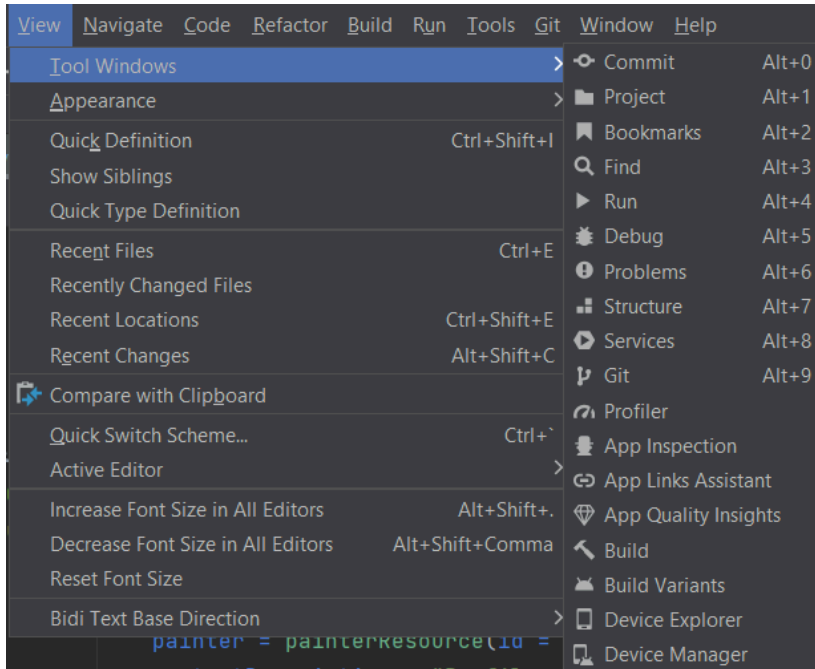
```
ProfilePic() {  
  
    Image(  
        painter = painterResource(id = R.drawable.brutus), // Replace with your image
```

Step 5: Set the content scale for the image to fit. Select and hover over the Image function to get the parameter name you need.

```
@Composable  
@ComposableTarget  
public fun Image(  
    painter: Painter,  
    contentDescription: String?,  
    modifier: Modifier = COMPILED_CODE,  
    alignment: Alignment = COMPILED_CODE,  
    contentScale: ContentScale = COMPILED_CODE,  
    alpha: Float = COMPILED_CODE,  
    colorFilter: ColorFilter? = COMPILED_CODE  
): Unit
```

Phase 6: Emulate your Profile.

Step 1: Make sure your Device Manager is open by using the drop-down menu: View>> Tool Windows>> Device Manager.

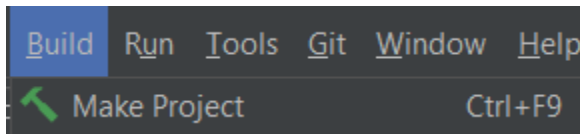


Step 2: Launch your device with the play button

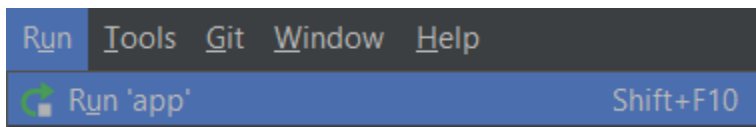


Step 3: Wait for the device to launch.

Step 4: Build your project using the drop-down menu: Build>>Make Project. Ctrl+F9 shortcut on Windows.



Step 5: Run your application on the device by using the drop-down menu: Run>> Run 'app'. Shift+F10 shortcut on Windows.



Phase 7: Edit YourCard in your profile and test

Step 1: Change the Card Title to "Button Card"

```
text = "Button Card",
```

Step 2: Remove the Text function that holds the Card Content. We will replace it with our new content.

```
Text(  
    text = "Card Content",  
    style = TextStyle(  
        // Add your own style  
    )  
)
```

Step 3: Add a Row function to replace the content.

```
Row() {  
    //Empty  
}
```

Step 4: Import the Row function by selecting and hovering over Row.

```
Row() {  
    //Unresolved reference: Row  
}
```

Import Alt+Shift+Enter More actions... Alt+Enter

Import the Composable Row function.


```
Row() {  
    //Imports  
}
```

@Composable Row(...) [...] (androidx.compose.foundation.layout) Gradle: androidx.compose.foundation:foundation-layout-android:1.5.0@aar (classes.jar)

Step 5: Add 3 buttons to the Row content as shown.

```
Row() { this: RowScope
    Button(onClick = { }) { this: RowScope
        Text( text: "Left")
    }
    Button(onClick = { }) { this: RowScope
        Text( text: "Middle")
    }
    Button(onClick = { }) { this: RowScope
        Text( text: "Right")
    }
}
```

Step 6: Emulate your new buttons by using the drop-down menu: Run>> Run 'app'

Notice this time the menu option icon has changed.  This will stop your application and rebuild it then start it with the new updates.

Also, Notice the buttons that you added do nothing. Verify they are displayed.

Step 7: Add context reference to the Row content. Copy the line of code as shown. Import the LocalContext property.

```
Row() { this: RowScope
    val context = LocalContext.current
```

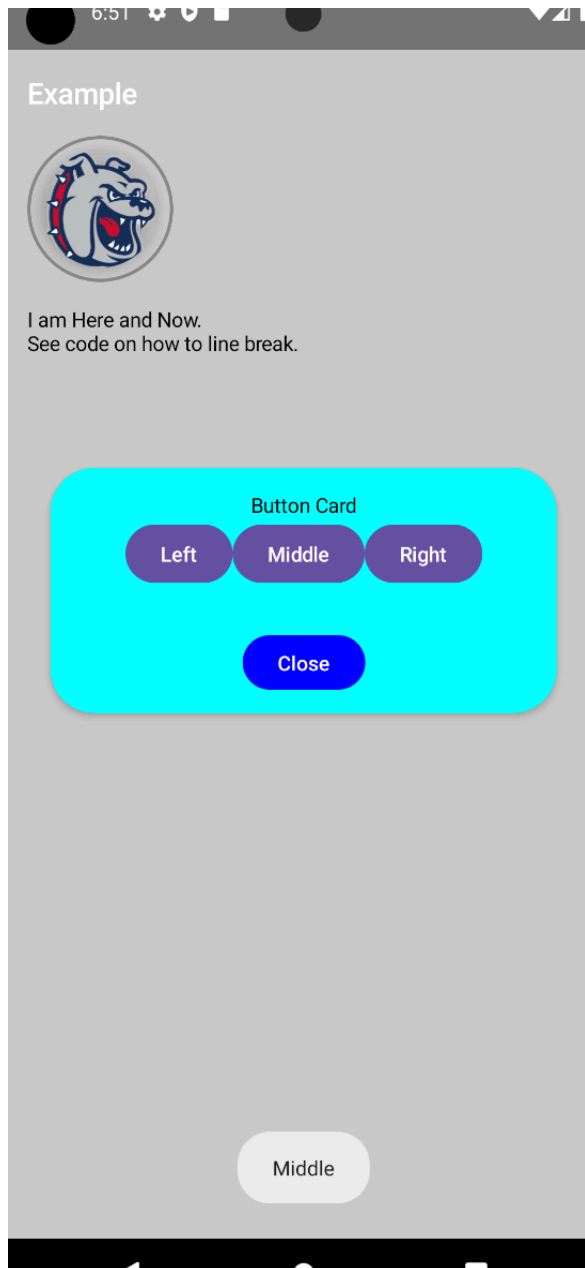
Step 8: Add a toast to your buttons. Use the line of code shown.

```
Toast.makeText(context, text: "Button Name", Toast.LENGTH_SHORT).show()
```

Add the Toast to the empty onClick body for each button. Rename the text to the actual button name.

```
Row() { this: RowScope
    val context = LocalContext.current
    Button(onClick = {Toast.makeText(context, text: "Left",Toast.LENGTH_SHORT).show()})
        Text( text: "Left")
    }
    Button(onClick = { }) { this: RowScope
        Text( text: "Middle")
    }
}
```

Step 9: Run your app and test your button.



Step Challenge: Add onClick actions to everything you can and make it do something. Use the Modifier object as shown to add a clickable action to a text.

```
Text(  
    modifier = Modifier.clickable { },
```

Try using a click action to change your profile name.
Change the background color.
Change your profile description.