

1

Creating Your First App

Activity 1.01 – Producing an app to create RGB colors Solution

Perform the following steps to solve the problem:

1. Create a new Android Studio project with an empty activity.
2. Update MainActivity to have a column with a title Text Composable:

```
Scaffold(  
    modifier = Modifier.fillMaxSize()  
) { innerPadding ->  
    Column(  
        horizontalAlignment =  
            Alignment.CenterHorizontally,  
        verticalArrangement =  
            Arrangement.spacedBy(16.dp),  
        modifier = Modifier  
            .fillMaxSize()  
            .padding(innerPadding)  
            .padding(16.dp)  
    ) {  
        Text("RGB Color Creator", fontSize = 24.sp)  
    }  
}
```

3. Add another Text Composable below with a brief description to the user on how to complete the form:

```
Text("Add two hexadecimal characters between 0-9, A-F or a-f without  
the '#' for each channel")
```

4. Add three MutableState properties for each of the three colors and another for a default color:

```
var redChannel by remember { mutableStateOf("") }  
var greenChannel by remember { mutableStateOf("") }  
var blueChannel by remember { mutableStateOf("") }  
var colorToDisplay by remember { mutableStateOf(Color.White) }  
}
```

5. Add three material OutlinedTextField buttons with labels of Red Channel, Green Channel, and Blue Channel, initialized with these MutableState properties:

```
OutlinedTextField(  
    modifier = Modifier.fillMaxWidth(),  
    value = redChannel,  
    onValueChange = { redChannel = it },  
    label = { Text("Red Channel") }  
)  
OutlinedTextField(  
    modifier = Modifier.fillMaxWidth(),  
    value = greenChannel,  
    onValueChange = { greenChannel = it },  
    label = { Text("Green Channel") }  
)  
OutlinedTextField(  
    modifier = Modifier.fillMaxWidth(),  
    value = blueChannel,  
    onValueChange = { blueChannel = it },  
    label = { Text("Blue Channel") }  
)
```

6. Add a function to restrict the input to only two hexadecimal characters. You can achieve this with the following function:

```
fun isValidHexInput(input: String): Boolean {
    return input.filter {
        it in '0'..'9' ||
        it in 'A'..'F' ||
        it in 'a'..'f'
    }.length == 2
}
```

7. Add a button that takes the inputs from the three color fields and uses the function:

```
Button(
    modifier = Modifier.fillMaxWidth(),
    onClick = {
        if (isValidHexInput(
            redChannel
        ) && isValidHexInput(
            greenChannel
        ) && isValidHexInput(
            blueChannel
        )
    ) {
        Text("CREATE COLOR")
    }
}
```

8. Update the button added in the preceding code to create a color string starting with the # character and then use Kotlin string templates to concatenate the colors together:

```
Button(
    modifier = Modifier.fillMaxWidth(),
    onClick = {
        if (isValidHexInput(
            redChannel
        ) && isValidHexInput(
            greenChannel
```

```

        ) && isValidHexInput(
            blueChannel
        )
    ) {
        val colorString =
            "$redChannel$greenChannel$blueChannel"
    }
}) {
    Text("CREATE COLOR")
}

```

9. Update it again to use a `Color` object to create a color from the string with `Color(colorString.toColorInt())`:

```

Button(
    modifier = Modifier.fillMaxWidth(),
    onClick = {
        if (isValidHexInput(
            redChannel
        ) && isValidHexInput(
            greenChannel
        ) && isValidHexInput(
            blueChannel
        )
    ) {
        val colorString =
            "$redChannel$greenChannel$blueChannel"
        colorToDisplay =
            Color(colorString.toColorInt())
    }
}) {
    Text("CREATE COLOR")
}

```

10. Finally, add another `Text` object to display the RGB color created from the three channels in the layout:

```

Text(
    modifier = Modifier

```

```
        .background(colorToDisplay)
        .padding(24.dp),
        text = "Created color display panel"
    )
```

The display of the app when run should look like the following or something similar:

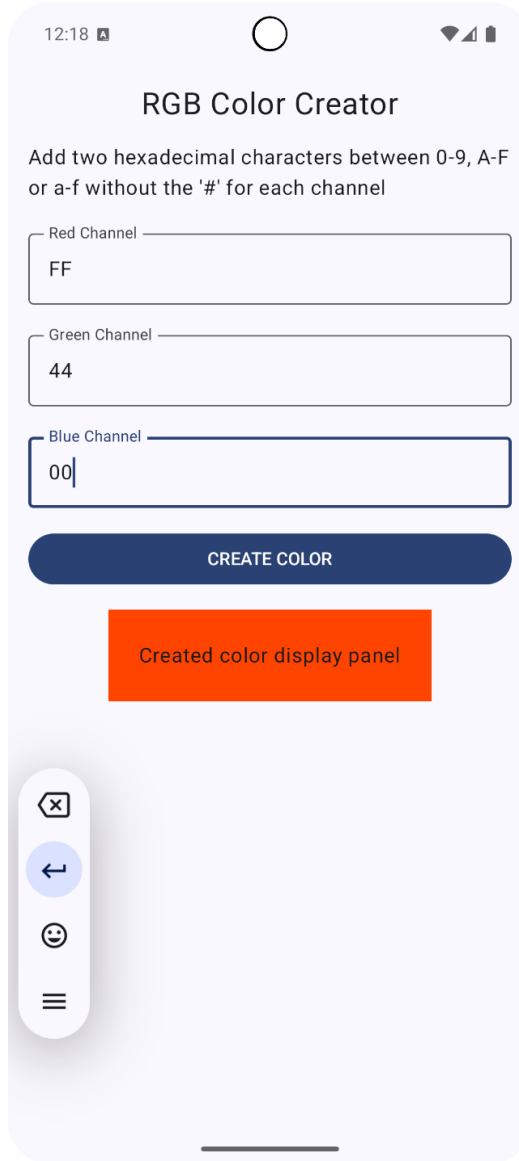


Figure 1.27 – RGB Color Creator with the created color displayed

