FR. CONCEICAO RODRIGUES COLLEGE OF ENGINEERING

Department of Computer Engineering

Class	**T.E. Computer A**
Subject Name	**Mobile Computing**

Practical No.	10
Title	Basic Android application primitives
Date of Performance	21/04/2025
Date of Submission	25/04/2025
Roll No.	9914
Name	Vivian

Evaluation:

Sr. No	Rubric	Grade
1	Timeline(2)	
2	Output(3)	
3	Code Optimization(3)	
4	Knowledge of the topic(2)	
5	Total(10)	

Signature of the teacher:

CODE:

```
//MainActivity.kt

package com.example.basicprimitives

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.background
import androidx.compose.foundation.border
import androidx.compose.foundation.clickable
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.size
import androidx.compose.foundation.layout.width
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.shape.CircleShape
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.foundation.verticalScroll
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.Add
import androidx.compose.material.icons.filled.Check
import androidx.compose.material.icons.filled.Favorite
import androidx.compose.material.icons.filled.Home
import androidx.compose.material3.Button
import androidx.compose.material3.Card
import androidx.compose.material3.CardDefaults
import androidx.compose.material3.Checkbox
import androidx.compose.material3.CircularProgressIndicator
import androidx.compose.material3.Divider
import androidx.compose.material3.ElevatedButton
import androidx.compose.material3.ExperimentalMaterial3Api
import androidx.compose.material3.FilledTonalButton
import androidx.compose.material3.FloatingActionButton
import androidx.compose.material3.Icon
import androidx.compose.material3.LinearProgressIndicator
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.OutlinedButton
import androidx.compose.material3.OutlinedTextField
import androidx.compose.material3.RadioButton
import androidx.compose.material3.Slider
import androidx.compose.material3.Surface
import androidx.compose.material3.Switch
import androidx.compose.material3.Text
import androidx.compose.material3.TextButton
import androidx.compose.material3.TextField
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.setValue
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.example.basicprimitives.ui.theme.BasicPrimitivesTheme

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            BasicPrimitivesTheme {
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    ComposeElementsShowcase()
                }
            }
        }
    }
}

@Composable
fun ComposeElementsShowcase() {
    Column(
        modifier = Modifier
            .fillMaxSize()
            .verticalScroll(rememberScrollState())
            .padding(16.dp),
        verticalArrangement = Arrangement.spacedBy(24.dp)
    ) {
        Text(
            text = "Jetpack Compose Primitives",
            style = MaterialTheme.typography.headlineMedium,
            fontWeight = FontWeight.Bold,
            modifier = Modifier.fillMaxWidth(),
            textAlign = TextAlign.Center
        )
```

```kotlin
        // Section: Text Examples
        SectionTitle("Text")
        TextExamples()

        // Section: Button Examples
        SectionTitle("Buttons")
        ButtonExamples()

        // Section: Layout Examples
        SectionTitle("Layouts")
        LayoutExamples()

        // Section: Input Controls
        SectionTitle("Input Controls")
        InputControls()

        // Section: Progress Indicators
        SectionTitle("Progress Indicators")
        ProgressIndicators()

        // Section: Images
        SectionTitle("Images")
        ImageExamples()

        // Section: Cards
        SectionTitle("Cards")
        CardExamples()

        // Space at the bottom
        Spacer(modifier = Modifier.height(32.dp))
    }
}

@Composable
fun SectionTitle(title: String) {
    Column {
        Text(
            text = title,
            style = MaterialTheme.typography.titleLarge,
            color = MaterialTheme.colorScheme.primary
        )
        Divider(
            modifier = Modifier.padding(vertical = 8.dp),
            color = MaterialTheme.colorScheme.primary.copy(alpha = 0.3f)
        )
    }
}

@Composable
fun TextExamples() {
    Column(verticalArrangement = Arrangement.spacedBy(8.dp)) {
        Text(text = "Default Text")
        Text(
            text = "Styled Text with Color and Size",
            color = Color.Blue,
            fontSize = 18.sp
        )
        Text(
            text = "Bold Text with Custom Weight",
            fontWeight = FontWeight.Bold
        )
        Text(
            text = "Text with Background",
            modifier = Modifier
                .background(Color.Yellow)
                .padding(4.dp)
        )
        Text(
            text = "Center Aligned Text with Max Width",
            textAlign = TextAlign.Center,
            modifier = Modifier.fillMaxWidth()
        )
    }
}

@Composable
fun ButtonExamples() {
    Column(verticalArrangement = Arrangement.spacedBy(8.dp)) {
        Button(onClick = { /* Do something */ }) {
            Text("Standard Button")
        }

        ElevatedButton(onClick = { /* Do something */ }) {
            Text("Elevated Button")
        }

        OutlinedButton(onClick = { /* Do something */ }) {
            Text("Outlined Button")
        }

        TextButton(onClick = { /* Do something */ }) {
            Text("Text Button")
        }

        FilledTonalButton(onClick = { /* Do something */ }) {
            Text("Filled Tonal Button")
        }

        Row(horizontalArrangement = Arrangement.spacedBy(8.dp)) {
            FloatingActionButton(
                onClick = { /* Do something */ },
                modifier = Modifier.size(56.dp)
            ) {
                Icon(Icons.Filled.Add, contentDescription = "Add")
            }

            FloatingActionButton(
                onClick = { /* Do something */ },
                containerColor = MaterialTheme.colorScheme.secondary,
                modifier = Modifier.size(56.dp)
            ) {
                Icon(Icons.Filled.Favorite, contentDescription = "Favorite")
            }
        }
    }
}

@Composable
fun LayoutExamples() {
    Column(verticalArrangement = Arrangement.spacedBy(16.dp)) {
        // Box example
        Text("Box (Stack items on top of each other):")
        Box(
            modifier = Modifier
                .fillMaxWidth()
                .height(100.dp)
                .background(Color.LightGray)
        ) {
            Box(
                modifier = Modifier
                    .size(60.dp)
                    .background(Color.Blue)
                    .align(Alignment.TopStart)
            )
            Box(
                modifier = Modifier
                    .size(60.dp)
                    .background(Color.Red)
                    .align(Alignment.Center)
            )
            Box(
                modifier = Modifier
                    .size(60.dp)
                    .background(Color.Green)
                    .align(Alignment.BottomEnd)
            )
        }

        // Row example
        Text("Row (Horizontal arrangement):")
        Row(
            modifier = Modifier
                .fillMaxWidth()
                .height(60.dp)
                .background(Color.LightGray),
            horizontalArrangement = Arrangement.SpaceEvenly,
            verticalAlignment = Alignment.CenterVertically
        ) {
            Box(modifier = Modifier.size(40.dp).background(Color.Red))
            Box(modifier = Modifier.size(40.dp).background(Color.Green))
            Box(modifier = Modifier.size(40.dp).background(Color.Blue))
        }

        // Column example
        Text("Column (Vertical arrangement):")
        Column(
            modifier = Modifier
                .fillMaxWidth()
                .height(150.dp)
                .background(Color.LightGray),
            verticalArrangement = Arrangement.SpaceEvenly,
            horizontalAlignment = Alignment.CenterHorizontally
        ) {
            Box(modifier = Modifier.size(40.dp).background(Color.Red))
            Box(modifier = Modifier.size(40.dp).background(Color.Green))
            Box(modifier = Modifier.size(40.dp).background(Color.Blue))
        }
    }
}

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun InputControls() {
    Column(verticalArrangement = Arrangement.spacedBy(16.dp)) {
        var text by remember { mutableStateOf("") }
        var checked by remember { mutableStateOf(false) }
        var selectedOption by remember { mutableStateOf(0) }
        var sliderPosition by remember { mutableStateOf(0.5f) }
        var switchChecked by remember { mutableStateOf(false) }

        // Text Field
        TextField(
            value = text,
            onValueChange = { text = it },
            label = { Text("Basic TextField") },
            modifier = Modifier.fillMaxWidth()
        )

        // Outlined Text Field
        OutlinedTextField(
            value = text,
            onValueChange = { text = it },
            label = { Text("Outlined TextField") },
            modifier = Modifier.fillMaxWidth()
        )

        // Checkbox
        Row(
            verticalAlignment = Alignment.CenterVertically,
            modifier = Modifier.fillMaxWidth()
        ) {
            Checkbox(
                checked = checked,
                onCheckedChange = { checked = it }
            )
            Spacer(modifier = Modifier.width(8.dp))
            Text("Checkbox")
        }

        // Radio Buttons
        Column {
            Text("Radio Buttons:")
            for (i in 0..2) {
                Row(
                    verticalAlignment = Alignment.CenterVertically
                ) {
                    RadioButton(
                        selected = selectedOption == i,
                        onClick = { selectedOption = i }
                    )
                    Spacer(modifier = Modifier.width(8.dp))
                    Text("Option ${i + 1}")
                }
            }
        }

        // Slider
        Column {
            Text("Slider: ${(sliderPosition * 100).toInt()}%")
            Slider(
                value = sliderPosition,
                onValueChange = { sliderPosition = it },
                modifier = Modifier.fillMaxWidth()
            )
        }

        // Switch
        Row(
            verticalAlignment = Alignment.CenterVertically,
            modifier = Modifier.fillMaxWidth()
        ) {
            Switch(
                checked = switchChecked,
                onCheckedChange = { switchChecked = it }
            )
            Spacer(modifier = Modifier.width(8.dp))
            Text("Switch")
        }
    }
```

```
}

@Composable
fun ProgressIndicators() {
    Column(verticalArrangement = Arrangement.spacedBy(16.dp)) {
        // Circular Progress Indicator
        Row(
            verticalAlignment = Alignment.CenterVertically,
            modifier = Modifier.fillMaxWidth()
        ) {
            CircularProgressIndicator()
            Spacer(modifier = Modifier.width(16.dp))
            Text("Circular Progress Indicator")
        }

        // Linear Progress Indicator
        Column {
            Text("Linear Progress Indicator")
            Spacer(modifier = Modifier.height(4.dp))
            LinearProgressIndicator(modifier = Modifier.fillMaxWidth())
        }

        // Determinate Progress Indicators
        Column {
            Text("Determinate Progress (70%)")
            Spacer(modifier = Modifier.height(4.dp))
            LinearProgressIndicator(
                progress = { 0.7f },
                modifier = Modifier.fillMaxWidth()
            )
            Spacer(modifier = Modifier.height(16.dp))
            CircularProgressIndicator(progress = { 0.7f })
        }
    }
}

@Composable
fun ImageExamples() {
    Column(verticalArrangement = Arrangement.spacedBy(16.dp)) {
        // Note: You need to have drawable resources in your project
        // For demonstration, we are using Icons
        Row(
            horizontalArrangement = Arrangement.spacedBy(16.dp),
            modifier = Modifier.fillMaxWidth()
        ) {
            // Icon example
            Column(
                horizontalAlignment = Alignment.CenterHorizontally
            ) {
                Icon(
                    imageVector = Icons.Default.Home,
                    contentDescription = "Home Icon",
                    modifier = Modifier.size(48.dp),
                    tint = MaterialTheme.colorScheme.primary
                )
                Spacer(modifier = Modifier.height(4.dp))
                Text("Icon")
            }

            // Circular clipped icon
            Column(
                horizontalAlignment = Alignment.CenterHorizontally
            ) {
                Icon(
                    imageVector = Icons.Default.Check,
                    contentDescription = "Check Icon",
                    modifier = Modifier
                        .size(48.dp)
                        .clip(CircleShape)
                        .background(MaterialTheme.colorScheme.primary)
                        .padding(8.dp),
                    tint = Color.White
                )
                Spacer(modifier = Modifier.height(4.dp))
                Text("Clipped Icon")
            }
        }

        Text("Note: For actual images, use the Image composable with your
resources:")
        Text("Image(painter = painterResource(id = R.drawable.your_image),
contentDescription = \"Image description\")")
    }
}

@Composable
fun CardExamples() {
    Column(verticalArrangement = Arrangement.spacedBy(16.dp)) {
        // Basic Card
        Card(
            modifier = Modifier.fillMaxWidth(),
            elevation = CardDefaults.cardElevation(defaultElevation = 4.dp)
        ) {
            Column(modifier = Modifier.padding(16.dp)) {
                Text(
                    text = "Basic Card",
                    style = MaterialTheme.typography.titleMedium
                )
                Spacer(modifier = Modifier.height(8.dp))
                Text(
                    text = "Cards contain content and actions about a single
subject."
                )
            }
        }

        // Outlined Card
        Card(
            modifier = Modifier.fillMaxWidth(),
            colors = CardDefaults.cardColors(
                containerColor = MaterialTheme.colorScheme.surface
            ),
            border = CardDefaults.outlinedCardBorder()
        ) {
            Column(modifier = Modifier.padding(16.dp)) {
                Text(
                    text = "Outlined Card",
                    style = MaterialTheme.typography.titleMedium
                )
                Spacer(modifier = Modifier.height(8.dp))
                Text(
                    text = "Cards with borders can be used for less elevated
content."
                )
            }
        }

        // Card with Rounded Corners
        Card(
            modifier = Modifier.fillMaxWidth(),
            shape = RoundedCornerShape(16.dp),
            colors = CardDefaults.cardColors(
                containerColor = MaterialTheme.colorScheme.primaryContainer
            )
        ) {
            Column(modifier = Modifier.padding(16.dp)) {
                Text(
                    text = "Rounded Card",
                    style = MaterialTheme.typography.titleMedium,
                    color = MaterialTheme.colorScheme.onPrimaryContainer
                )
                Spacer(modifier = Modifier.height(8.dp))
                Text(
                    text = "This card has more rounded corners and a custom
background color.",
                    color = MaterialTheme.colorScheme.onPrimaryContainer
                )
            }
        }

        // Clickable Card
        var clickCount by remember { mutableStateOf(0) }
        Card(
            modifier = Modifier
                .fillMaxWidth()
                .clickable { clickCount++ },
            elevation = CardDefaults.cardElevation(defaultElevation = 4.dp)
        ) {
            Column(modifier = Modifier.padding(16.dp)) {
                Text(
                    text = "Clickable Card",
                    style = MaterialTheme.typography.titleMedium
                )
                Spacer(modifier = Modifier.height(8.dp))
                Text(
                    text = if (clickCount > 0) "You clicked this card
$clickCount times" else "Click this card!"
                )
            }
        }
    }
}

@Preview(showBackground = true)
@Composable
fun PreviewComposeElementsShowcase() {
    BasicPrimitivesTheme {
        Surface(
            modifier = Modifier.fillMaxSize(),
            color = MaterialTheme.colorScheme.background
        ) {
            ComposeElementsShowcase()
        }
    }
}
```

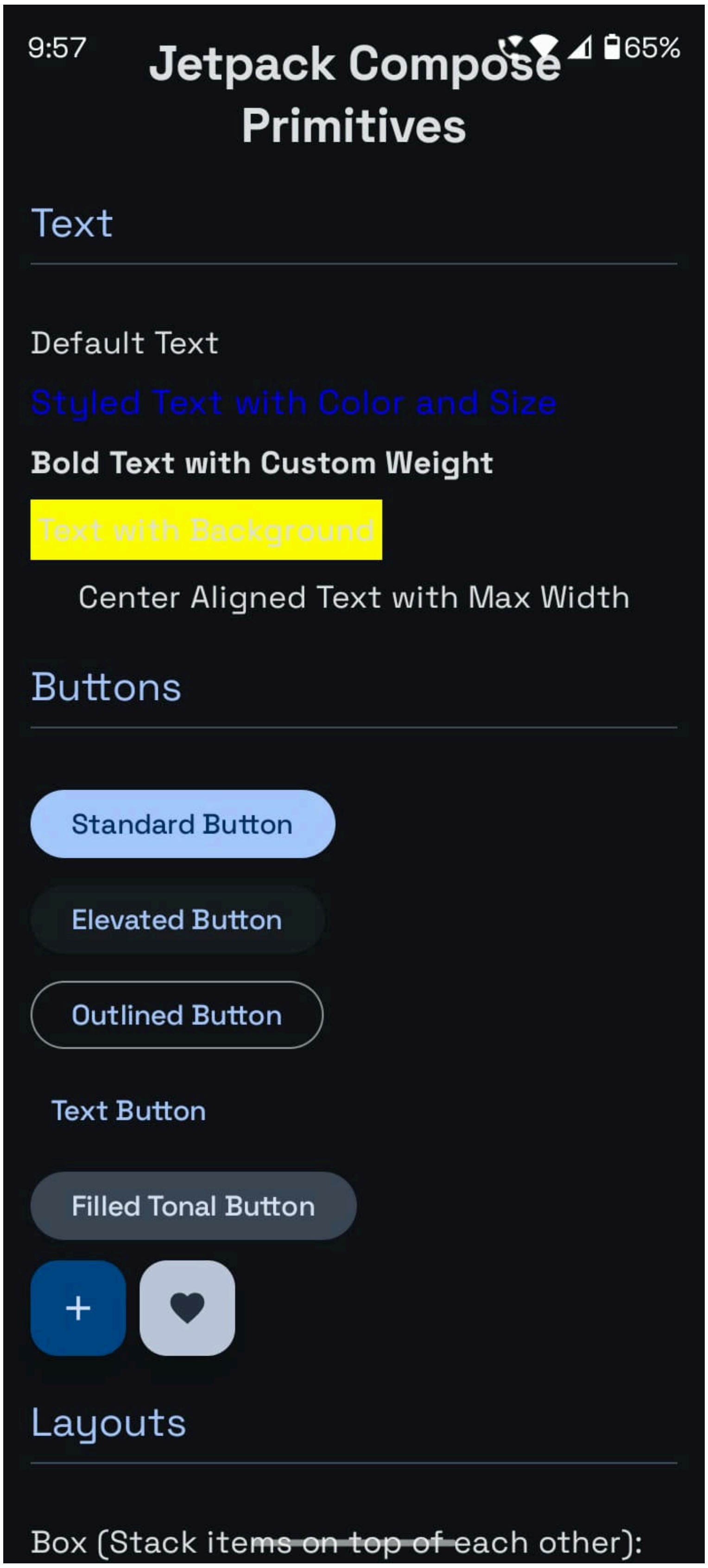
```
//libs.versions.toml
```

```
[versions]
agp = "8.9.1"
kotlin = "2.0.21"
coreKtx = "1.10.1"
junit = "4.13.2"
junitVersion = "1.1.5"
espressoCore = "3.5.1"
lifecycleRuntimeKtx = "2.6.1"
activityCompose = "1.8.0"
composeBom = "2024.09.00"

[libraries]
androidx-core-ktx = { group = "androidx.core", name = "core-ktx", version.ref
= "coreKtx" }
junit = { group = "junit", name = "junit", version.ref = "junit" }
androidx-junit = { group = "androidx.test.ext", name = "junit", version.ref =
"junitVersion" }
androidx-espresso-core = { group = "androidx.test.espresso", name = "espresso-
core", version.ref = "espressoCore" }
androidx-lifecycle-runtime-ktx = { group = "androidx.lifecycle", name =
"lifecycle-runtime-ktx", version.ref = "lifecycleRuntimeKtx" }
androidx-activity-compose = { group = "androidx.activity", name = "activity-
compose", version.ref = "activityCompose" }
androidx-compose-bom = { group = "androidx.compose", name = "compose-bom",
version.ref = "composeBom" }
androidx-ui = { group = "androidx.compose.ui", name = "ui" }
androidx-ui-graphics = { group = "androidx.compose.ui", name = "ui-graphics" }
androidx-ui-tooling = { group = "androidx.compose.ui", name = "ui-tooling" }
androidx-ui-tooling-preview = { group = "androidx.compose.ui", name = "ui-
tooling-preview" }
androidx-ui-test-manifest = { group = "androidx.compose.ui", name = "ui-test-
manifest" }
androidx-ui-test-junit4 = { group = "androidx.compose.ui", name = "ui-test-
junit4" }
androidx-material3 = { group = "androidx.compose.material3", name =
"material3" }

[plugins]
android-application = { id = "com.android.application", version.ref = "agp" }
kotlin-android = { id = "org.jetbrains.kotlin.android", version.ref =
"kotlin" }
kotlin-compose = { id = "org.jetbrains.kotlin.plugin.compose", version.ref =
"kotlin" }
```

```
//build.gradle.kts
```

```
plugins {
    alias(libs.plugins.android.application)
    alias(libs.plugins.kotlin.android)
    alias(libs.plugins.kotlin.compose)
}

android {
    namespace = "com.example.basicprimitives"
    compileSdk = 35

    defaultConfig {
        applicationId = "com.example.basicprimitives"
        minSdk = 24
        targetSdk = 35
        versionCode = 1
        versionName = "1.0"

        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }
    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_11
        targetCompatibility = JavaVersion.VERSION_11
    }
    kotlinOptions {
        jvmTarget = "11"
    }
    buildFeatures {
        compose = true
    }
}

dependencies {

    implementation(libs.androidx.core.ktx)
    implementation(libs.androidx.lifecycle.runtime.ktx)
    implementation(libs.androidx.activity.compose)
    implementation(platform(libs.androidx.compose.bom))
    implementation(libs.androidx.ui)
    implementation(libs.androidx.ui.graphics)
    implementation(libs.androidx.ui.tooling.preview)
    implementation(libs.androidx.material3)
    testImplementation(libs.junit)
    androidTestImplementation(libs.androidx.junit)
    androidTestImplementation(libs.androidx.espresso.core)
    androidTestImplementation(platform(libs.androidx.compose.bom))
    androidTestImplementation(libs.androidx.ui.test.junit4)
    debugImplementation(libs.androidx.ui.tooling)
    debugImplementation(libs.androidx.ui.test.manifest)
}
```

OUTPUT:



9:57 65%

Jetpack Compose Primitives

Text

Default Text

Styled Text with Color and Size

Bold Text with Custom Weight

Text with Background

Center Aligned Text with Max Width

Buttons

Standard Button

Elevated Button

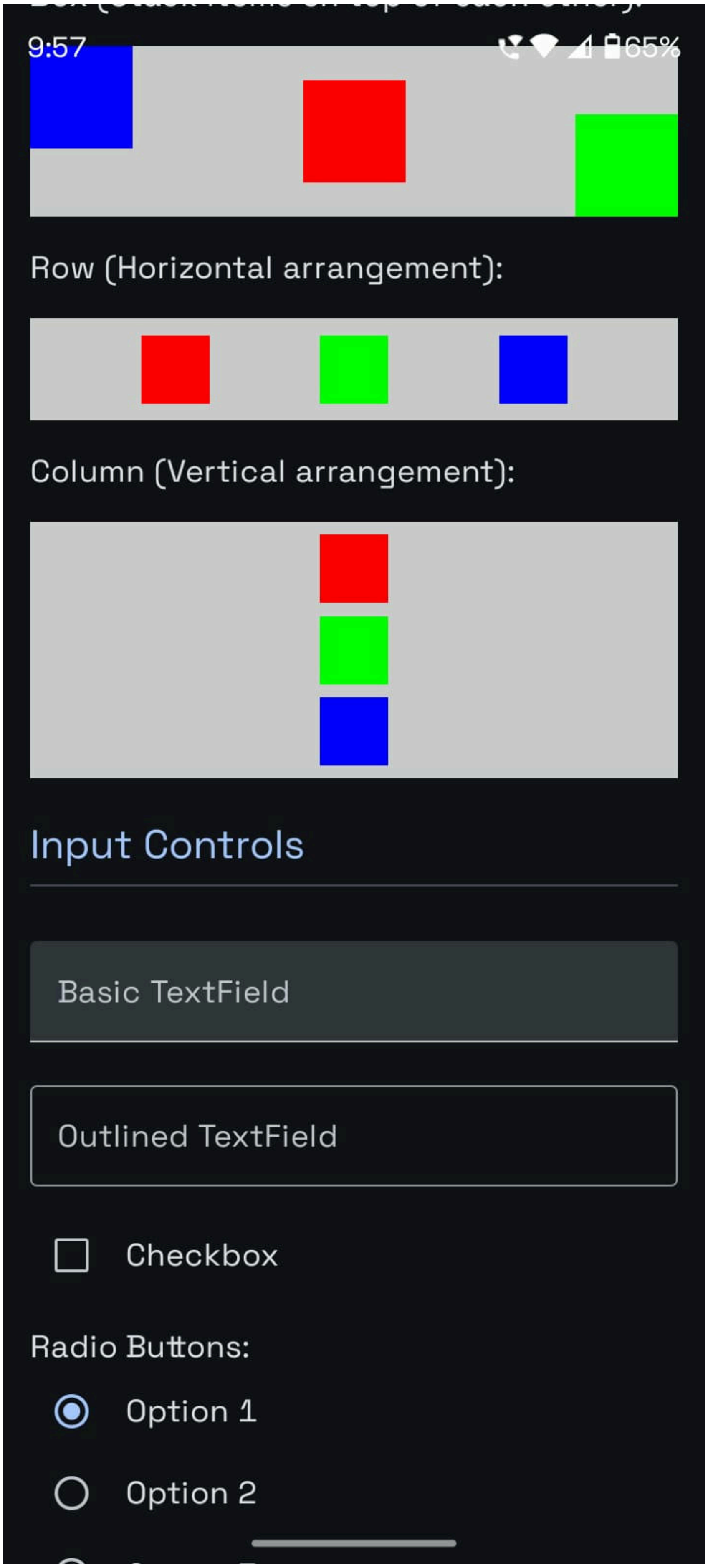
Outlined Button

Text Button

Filled Tonal Button

Layouts

Box (Stack items on top of each other):


9:57 65%

Row (Horizontal arrangement):

Column (Vertical arrangement):

Input Controls

Basic TextField

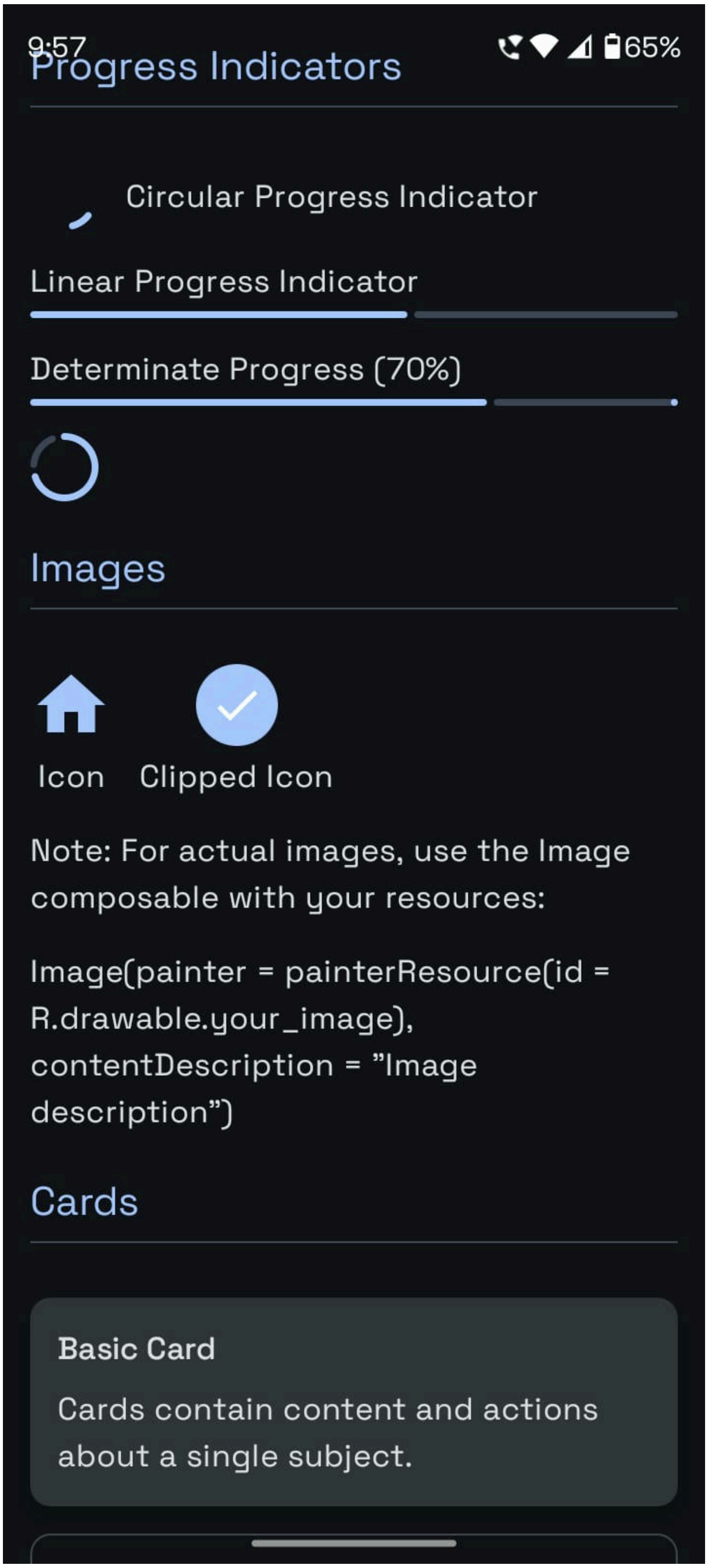
Outlined TextField

☐ Checkbox

Radio Buttons:

● Option 1

○ Option 2



9:57 65%

Progress Indicators

Circular Progress Indicator

Linear Progress Indicator

Determinate Progress (70%)

Images

Icon Clipped Icon

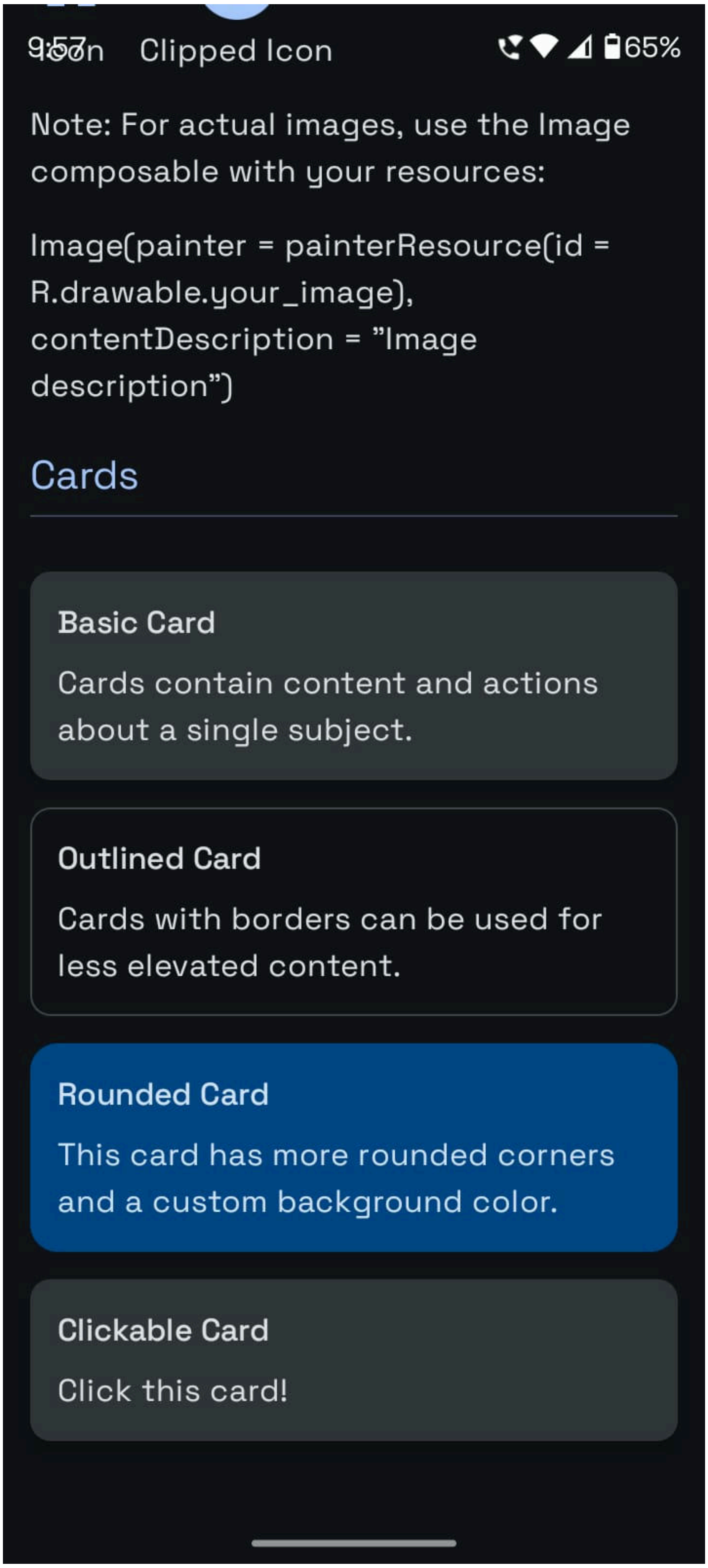
Note: For actual images, use the Image composable with your resources:

Image(painter = painterResource(id = R.drawable.your_image), contentDescription = "Image description")

Cards

Basic Card

Cards contain content and actions about a single subject.


9:57n Clipped Icon 65%

Note: For actual images, use the Image composable with your resources:

Image(painter = painterResource(id = R.drawable.your_image), contentDescription = "Image description")

Cards

Basic Card

Cards contain content and actions about a single subject.

Outlined Card

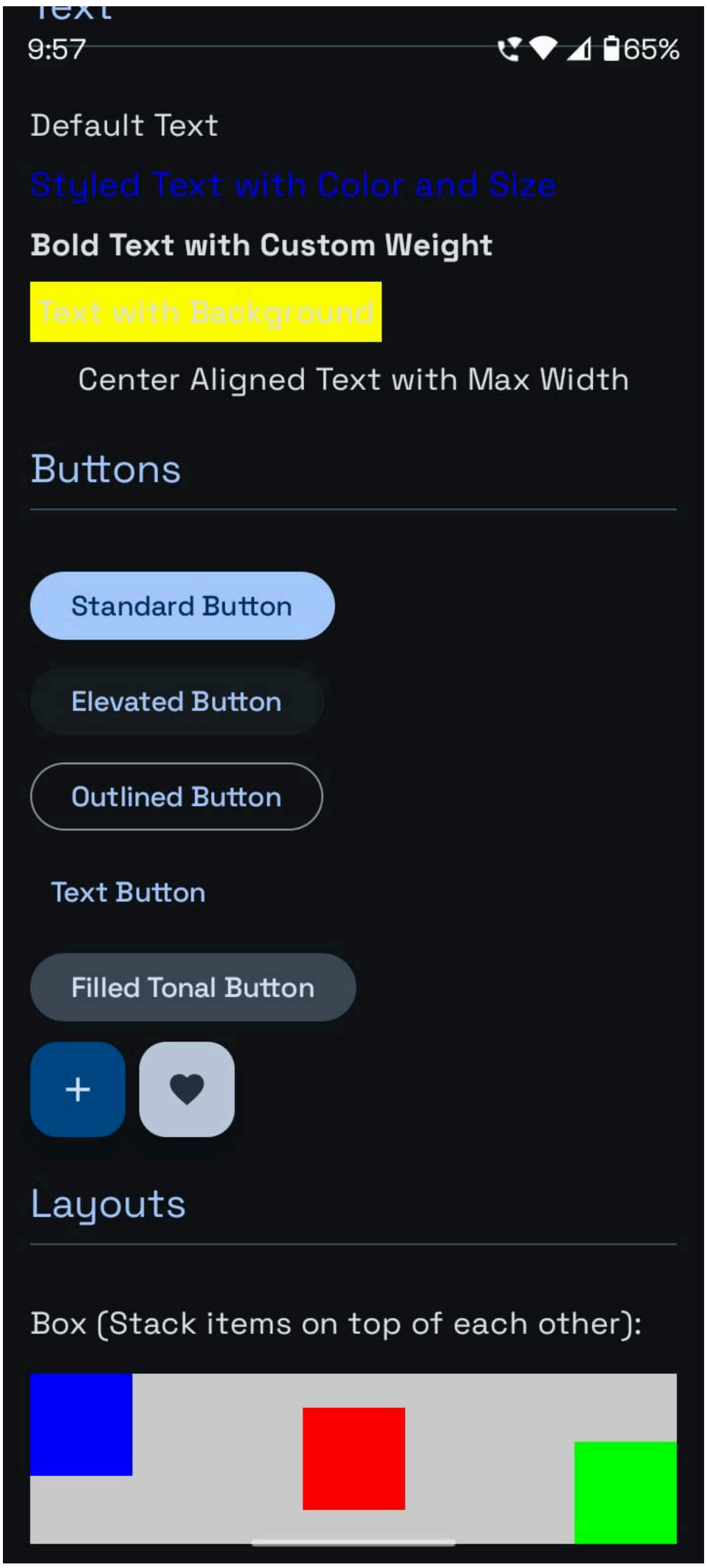
Cards with borders can be used for less elevated content.

Rounded Card

This card has more rounded corners and a custom background color.

Clickable Card

Click this card!


9:57
65%
Default Text
Styled Text with Color and Size
Bold Text with Custom Weight
Text with Background
Center Aligned Text with Max Width
Buttons
Standard Button
Elevated Button
Outlined Button
Text Button
Filled Tonal Button
+
Layouts
Box (Stack items on top of each other):