



Module Code & Title

CS6P05 Final Year Project MAD

Food Share - Android App

Artifact – Mobile UI development

Student Details

Name: Sita Ram Thing

London Met Id: 22015892

College Id: NP01MA4S220003

Islington College, Kathmandu

24 April 2024

S

1 Introduction

1.1 Material design

What's Material?

Material Design is a design system built and supported by Google designers and developers. **Material.io** includes in-depth UX guidance and UI component implementations for Android, Flutter, and the Web.

The latest version, Material 3, enables personal, adaptive, and expressive experiences – from dynamic color and enhanced accessibility, to foundations for large screen layouts and design tokens.

UX foundations



Foundations like color, type, and shape are customizable systems in Material

Open-source code



Multi-platform code to build beautiful products, faster

Figure 1: Material introduction

The 1.2 release of Compose Material 3 is here, and with it comes new components, some component changes and an expansion of the Material3 color system.

Component Changes, Demotions and Promotions

`Segmented Button` is a new experimental component. There are single select and multiple selection variants.

Figure 2: Material changes and promotions.

Component Changes, Demotions and Promotions

Segmented Button is a new experimental component. There are single select and multiple selection variants.

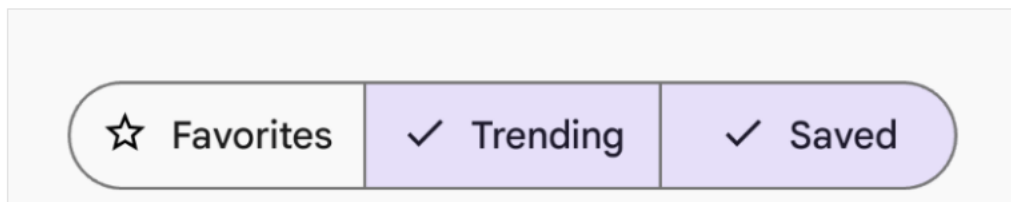
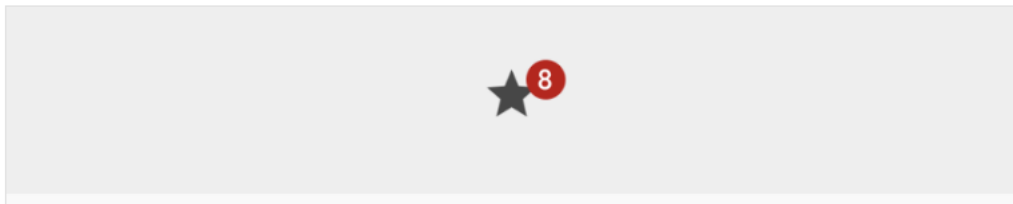


Figure 3: Material component

`BottomAppBar` has a `BottomAppBarScrollBehavior` to auto-hide itself when content is scrolled.

`SwipeToDismiss` has been refactored into `SwipeDismissBox` and remains in experimental status.

`Badge` and `BadgedBox` have been promoted to stable.



The `Chip` APIs have been promoted to stable.

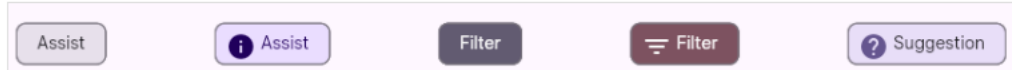


Figure 4: BottomAppBar and badge box

On Primary	P-100	On Secondary	S-100	On Tertiary	T-100	On Error	E-100
Primary Container		Secondary Container		Tertiary Container		Error Container	
	P-90		S-90		T-90		E-90
On Primary Container	P-10	On Secondary Container	S-10	On Tertiary Container	T-10	On Error Container	E-10
Surface Dim		Surface		Surface Bright		Inverse Surface	
	N-87		N-98		N-98		N-20
Surf. Container Lowest		Surf. Container Low		Surf. Container		Surf. Container High	
	N-100		N-96		N-94		N-92
							N-90
On Surface	N-10	On Surface Var.	NV-30	Outline	NV-50	Outline Variant	NV-80
						Scrim	N-0
						Shadow	N-0

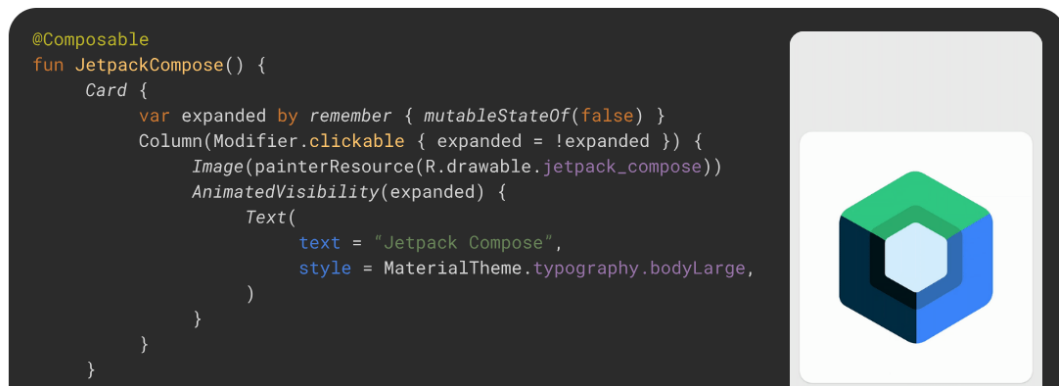
Primary		Secondary		Tertiary		Error	
	P-80		S-80		T-80		E-80
On Primary	P-20	On Secondary	S-20	On Tertiary	T-20	On Error	E-20
Primary Container		Secondary Container		Tertiary Container		Error Container	
	P-30		S-30		T-30		E-30
On Primary Container	P-90	On Secondary Container	S-90	On Tertiary Container	T-90	On Error Container	E-90
Surface Dim		Surface		Surface Bright		Inverse Surface	
	N-6		N-6		N-24		N-90
Surf. Container Lowest		Surf. Container Low		Surf. Container		Surf. Container High	
	N-4		N-10		N-12		N-17
							N-24
On Surface	N-90	On Surface Var.	NV-90	Outline	NV-60	Outline Variant	NV-30
						Scrim	N-0
						Shadow	N-0

Figure 5: Material colour schema.

1.2 Jet pack compose.

Build better apps faster with Jetpack Compose

Jetpack Compose is Android's recommended modern toolkit for building native UI. It simplifies and accelerates UI development on Android. Quickly bring your app to life with less code, powerful tools, and intuitive Kotlin APIs.



Set up Compose for an existing app

To start using Compose, you need to first add some build configurations to your project. Add the following definition to your app's `build.gradle` file:

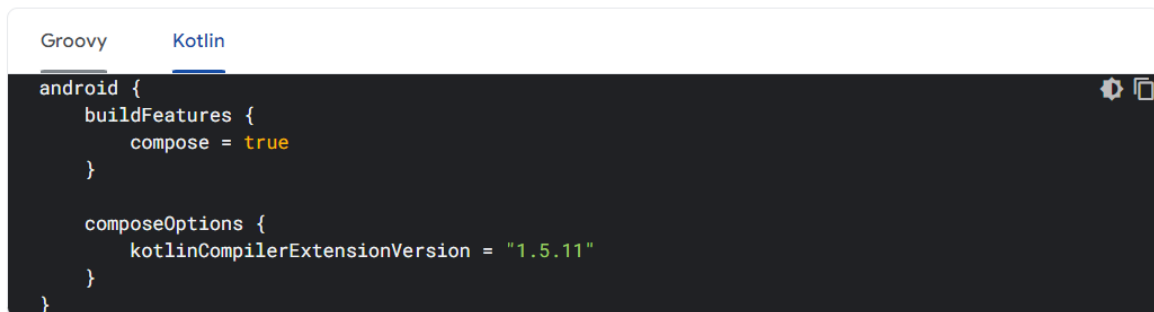


Figure 6: set up the compose in Kotlin.


```
dependencies {

    val composeBom = platform("androidx.compose:compose-bom:2024.04.01")
    implementation(composeBom)
    androidTestImplementation(composeBom)

    // Choose one of the following:
    // Material Design 3
    implementation("androidx.compose.material3:material3")
    // or Material Design 2
    implementation("androidx.compose.material:material")
    // or skip Material Design and build directly on top of foundational components
    implementation("androidx.compose.foundation:foundation")
    // or only import the main APIs for the underlying toolkit systems,
    // such as input and measurement/layout
    implementation("androidx.compose.ui:ui")

    // Android Studio Preview support
    implementation("androidx.compose.ui:ui-tooling-preview")
    debugImplementation("androidx.compose.ui:ui-tooling")

    // UI Tests
    androidTestImplementation("androidx.compose.ui:ui-test-junit4")
    debugImplementation("androidx.compose.ui:ui-test-manifest")

    // Optional - Included automatically by material, only add when you need
    // the icons but not the material library (e.g. when using Material3 or a
    // custom design system based on Foundation)
    implementation("androidx.compose.material:material-icons-core")
    // Optional - Add full set of material icons
    implementation("androidx.compose.material:material-icons-extended")
    // Optional - Add window size utils
    implementation("androidx.compose.material3:material3-window-size-class")
}
```

Figure 7: dependency implementations.

1.3 Implementation of the compose

```
// compose
implementation("androidx.compose.material:material:1.6.5")
debugImplementation("androidx.compose.ui:ui-tooling:1.6.5")
implementation("androidx.compose.ui:ui-tooling-preview:1.6.5")
implementation("androidx.compose.runtime:runtime-livedata:1.6.5")
implementation("androidx.navigation:navigation-fragment-ktx:2.7.7")
implementation("androidx.navigation:navigation-ui-ktx:2.7.7")

// activity
implementation("androidx.activity:activity-compose:1.8.2")

// Lifecycle
implementation("androidx.lifecycle:lifecycle-viewmodel-ktx:2.7.0")
implementation("androidx.lifecycle:lifecycle-livedata-ktx:2.7.0")
implementation("androidx.lifecycle:lifecycle-common:2.7.0")
implementation("androidx.lifecycle:lifecycle-runtime-ktx:2.7.0")
implementation("androidx.lifecycle:lifecycle-extensions:2.2.0")

// ViewModel and LiveData for Compose
implementation("androidx.lifecycle:lifecycle-viewmodel-compose:2.7.0")

// tab layout
implementation("com.google.accompanist:accompanist-pager:0.28.0")
```

Figure 8: Impalement of the dependency.

👤 Sita Ram Thing

```
@SuppressLint("MissingPermission")
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    // A surface container using the 'background' color from the theme

    setContent {
        FoodShareTheme {
            Surface(
                modifier = Modifier.fillMaxSize(),
                color = backgroundLayoutColor // MaterialTheme.colorScheme.background
            ) {
                val context = LocalContext.current
                val getPreInstance = UserInterceptors(context)
                val systemToken = getPreInstance.getSystemToken()
                val accessToken = getPreInstance.getAccessToken()
                if (accessToken.isNotEmpty() && systemToken.isNotEmpty()) {
                    if (isTokenExpired(accessToken, context, systemToken)) {
                        showToast(context, "Authentication token is expired.")
                    }
                }
            }
            MainNavigationViewScreen()
        }
    }
}
```

Figure 9: Set up the composed video in the main activity.

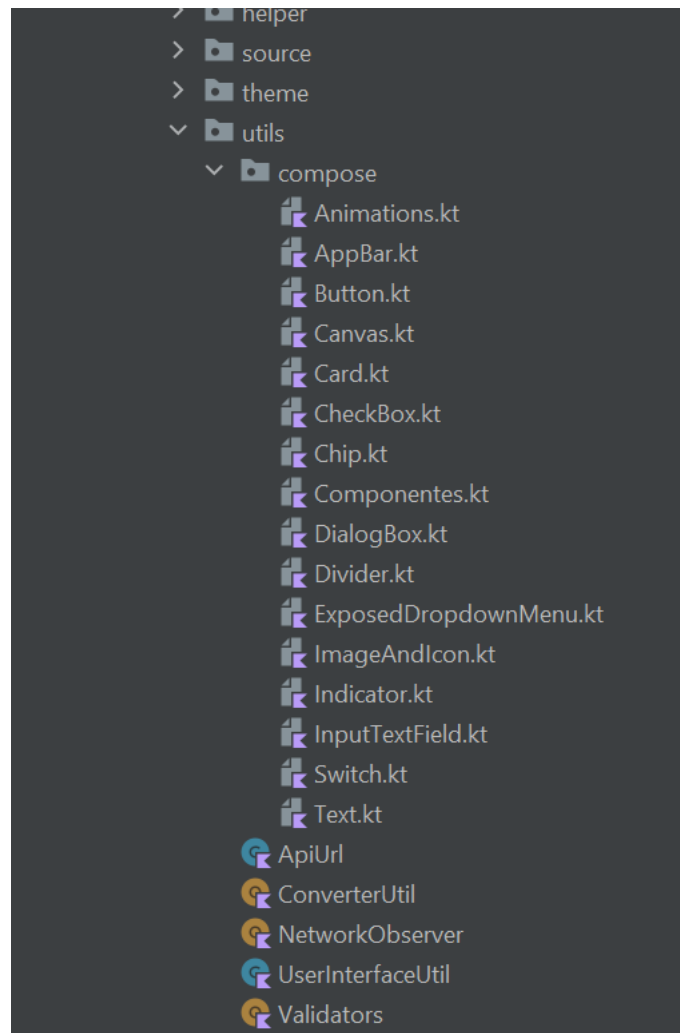


Figure 10: Create the custom component

```

onClick: () -> Unit,
modifier: Modifier = Modifier,
colors: ButtonColors = ButtonDefaults.buttonColors(primaryColor, disabledColor),
btnText: String,
shape: Shape = ShapeDefaults.ExtraLarge,
textType: TextType = TextType.BUTTON_TEXT_REGULAR,
enabled: Boolean = true,
elevation: ButtonElevation? = null,
border: BorderStroke? = null,
contentPadding: PaddingValues = PaddingValues(0.dp),
textColors: Color = white,
) {
    Button(
        onClick = { onClick() },
        modifier = modifier,
        colors = colors,
        enabled = enabled,
        elevation = elevation,
        shape = shape,
        border = border,
        contentPadding = contentPadding
    ) {

```

Figure 11: Implement the compose button