LONDON METROPOLITAN UNIVERSITY

Islington college
(इस्लिङ्टन कलेज)

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

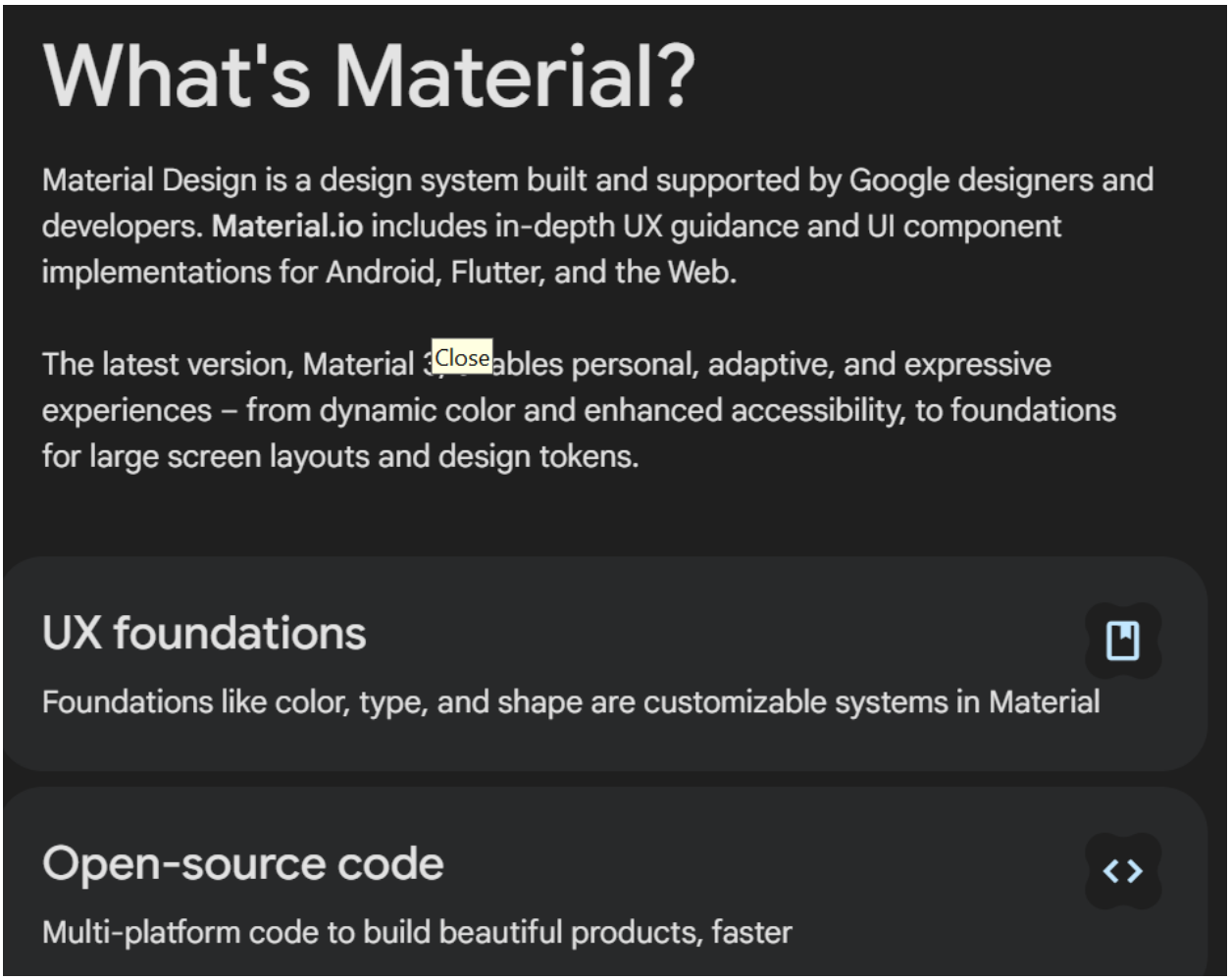


Figure 1: Material introduction

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.

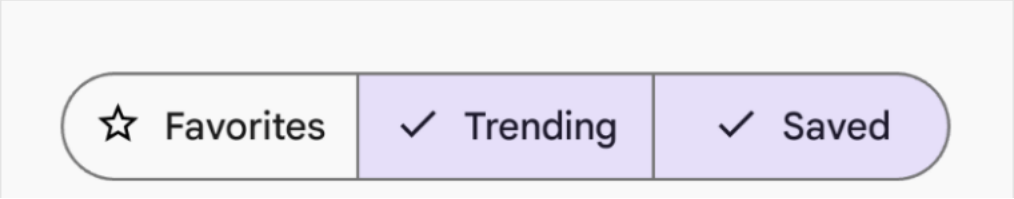
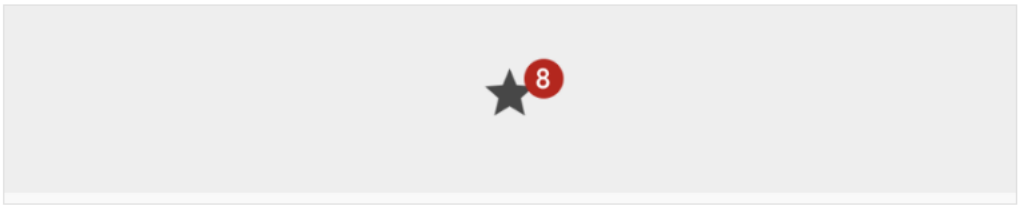


Figure 3: Material component

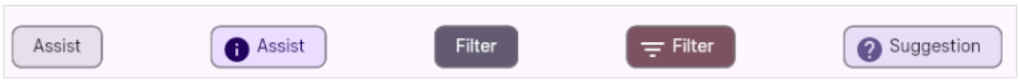
`BottomAppBar` has a `BottomAppBarScrollBehavior` to auto-hide itself when content is scrolled.

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

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

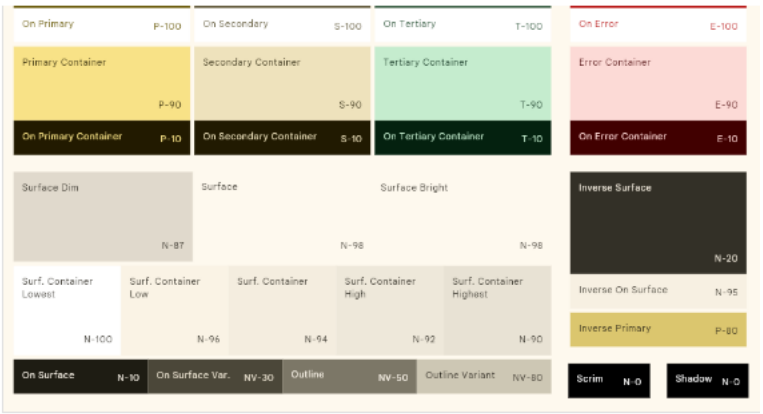


The `Chip` APIs have been promoted to stable.



\

Figure 4: BottomAppBaran and badge box



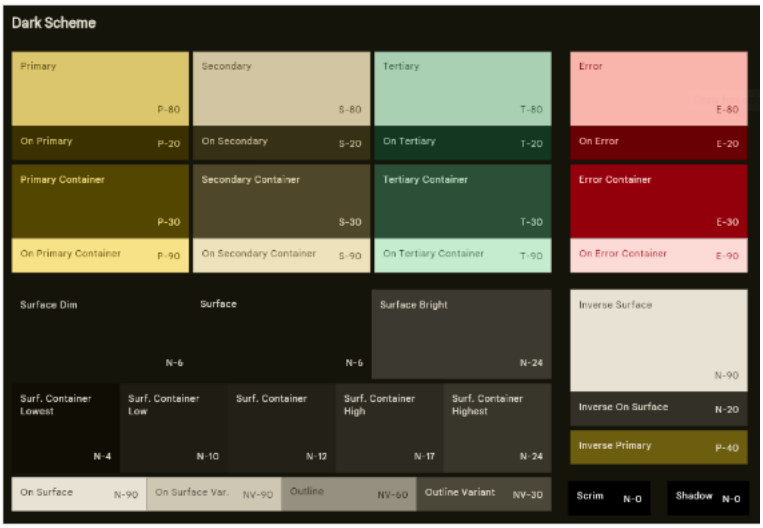
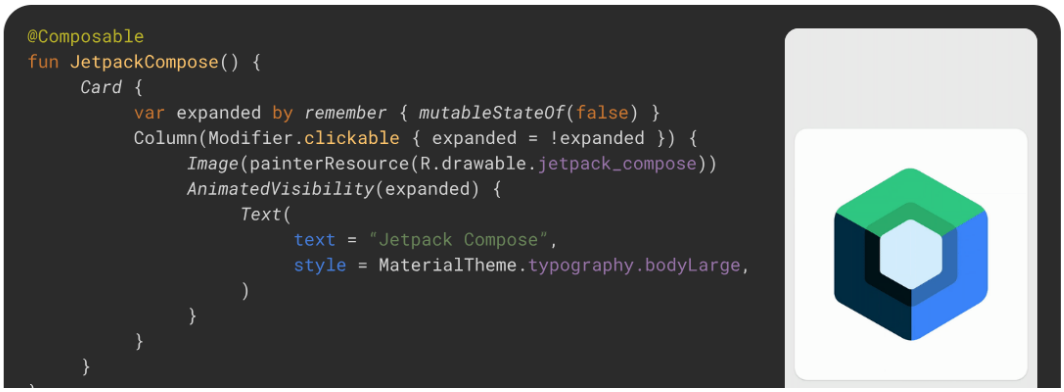


Figure 5: Material colour schema.

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:

Groovy Kotlin

```
android {
    buildFeatures {
        compose = true
    }

    composeOptions {
        kotlinCompilerExtensionVersion = "1.5.11"
    }
}
```

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 impalements.

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.

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

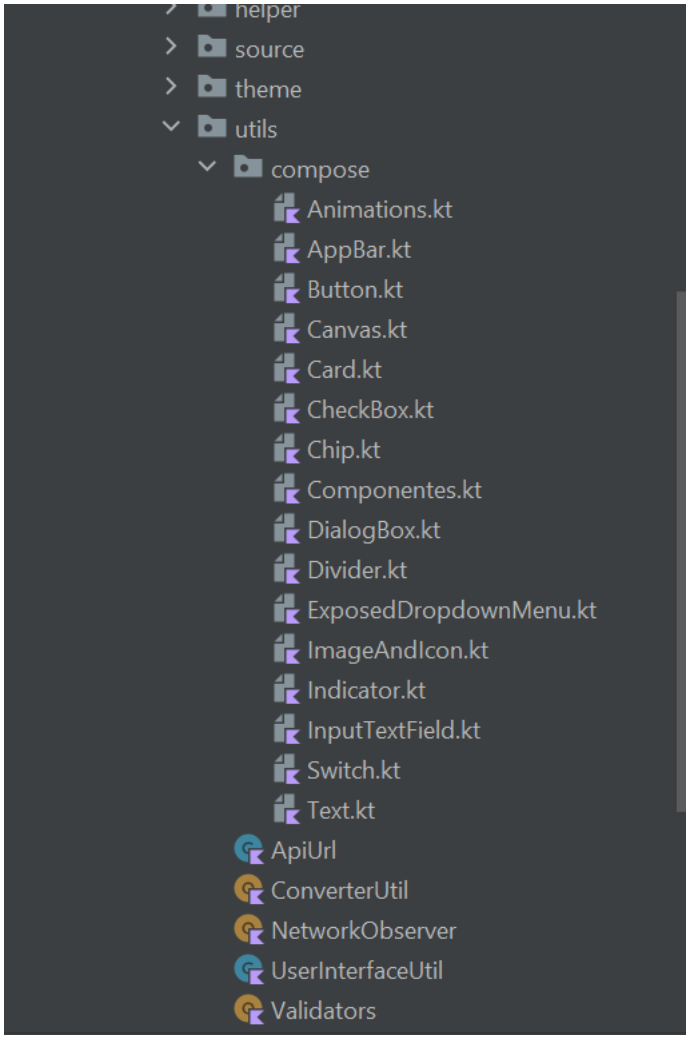


Figure 10: Create the custom component

```
    onClick: () -> Unit,
    modifier: Modifier = Modifier,
    colors: ButtonColors = ButtonDefaults.buttonColors(prima
    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