JETPACK COMPOSE UI DESIGN CHEATSHEET

Scaffold and TopAppBar

Scaffold:

Provides a basic structure for app UI.

TopAppBar:

Contains title and action icons.

```
val itemTouchHelperCallback = object :
ItemTouchHelper.Sim
pleCallback(0, ItemTouchHelper.LEFT) {
override fun onSwiped(viewHolder:
RecyclerView.ViewHol
der, direction: Int) {
// Handle swipe action
}
}
ItemTouchHelper(itemTouchHelperCallback)
.attachToRecyclerV
iew(recyclerView)
```

LazyRow in Jetpack Compose

LazyRow:

Horizontally scrollable list.

```
@Composable
fun MyLazyRow() {
    LazyRow {
        items(listOf("Item 1",
"Item 2", "Item 3")) { item ->
            Text(item)
        }
    }
}
```

Vector Assets and Asset Studio

Vector Assets:

Scalable graphics in XML format.

Asset Studio:

Tool in Android Studio for creating and managing assets.

Import Vector Asset:

File → New → Vector Asset.

Customize and save in res/drawable .

Usage:

```
Icon(painter =
painterResource(R.drawable.ic_vector), c
ontentDescription = "Icon")
```

ModalBottomSheetLayout in Jetpack Compose

ModalBottomSheetLayout:

Sliding up from the bottom, blocking interaction.

```
@Composable
fun MyScreen() {
    val sheetState =
rememberModalBottomSheetState(
initialValue =
ModalBottomSheetValue.Hidden)
    ModalBottomSheetLayout(
        sheetState =
sheetState,
        sheetContent = {
Text("Here's some content in
the bottom sheet") }
    ) {
        Button(onClick = {
sheetState.show() }) {
            Text("Show Bottom
Sheet")
        }
    }
}
```

Sticky Header in Jetpack Compose

Sticky Header:

Stays at the top while scrolling.

```
@Composable
fun StickyHeaderTextExample() {
LazyColumn {
stickyHeader { Text("Sticky Header",
fontSize = 2
0.sp) }
items(listOf("Item 1", "Item 2", "Item
3")) { item
→
Text(item, fontSize = 16.sp)
}
}
}
```

Icon Tint Property in Jetpack Compose

Tinting:

Applying color filter to icons.

```
@Composable
fun TintedIcon() {
Icon(
painter =
painterResource(R.drawable.ic_icon),
contentDescription = "Icon",
tint = Color.Red
)
}
```

Dynamic Tint:

```
val isSelected = remember {
mutableStateOf(false) }
Icon(
painter =
painterResource(R.drawable.ic_icon),
contentDescription = "Icon",
tint = if (isSelected.value) Color.Green
else Color.Gr
ay
)
```