Scaffold & Navigation Menu

Asst. Prof. Monlica Wattana, Ph.D
Department of Computer Science,
Khon Kaen University

SC 362 007 Mobile Device Programming

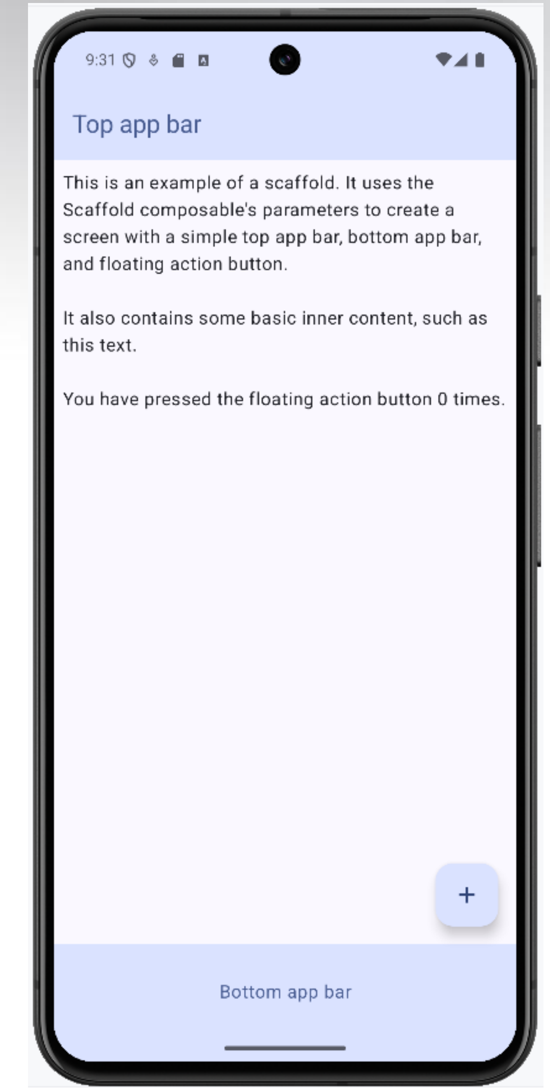
CP 410 804 Programming for Mobile Application

Outline

- Scaffold
- Top App Bar Navigation
  - Menu
  - Navigation Drawer
- Bottom Navigation Bar
- Scaffold & Navigation Components

Scaffold

- A scaffold is a fundamental structure that provides a standardized platform for complex user interfaces.
- It holds together different parts of the UI, such as app bars and floating action buttons, giving apps a coherent look and feel.



This is an example of a scaffold. It uses the Scaffold composable's parameters to create a screen with a simple top app bar, bottom app bar, and floating action button.

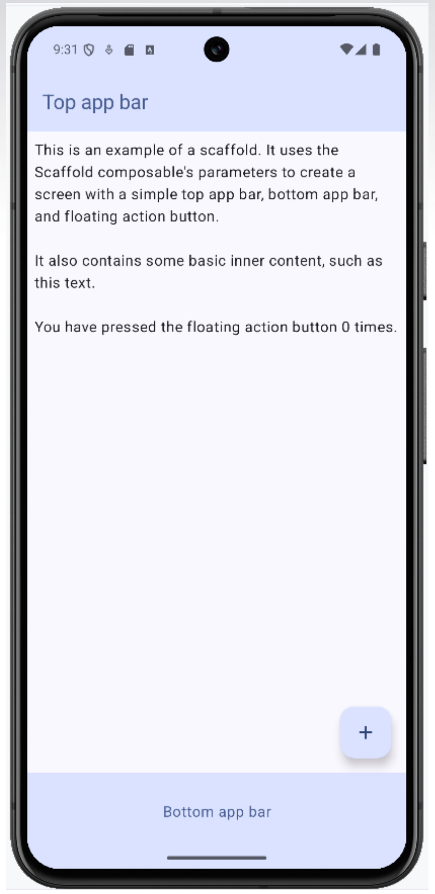
It also contains some basic inner content, such as this text.

You have pressed the floating action button 0 times.

Scaffold

Scaffold Components

- topBar: The header slot for the app bar, displaying the title and navigation actions.
- bottomBar: The footer slot, ideal for placing a bottom navigation bar.
- floatingActionButton (FAB): A floating button reserved for the primary action of the screen.
- content: The main display area (Body) where the screen's content is placed (Requires handling innerPadding to prevent overlap).



Top app bar

This is an example of a scaffold. It uses the Scaffold composable's parameters to create a screen with a simple top app bar, bottom app bar, and floating action button.

It also contains some basic inner content, such as this text.

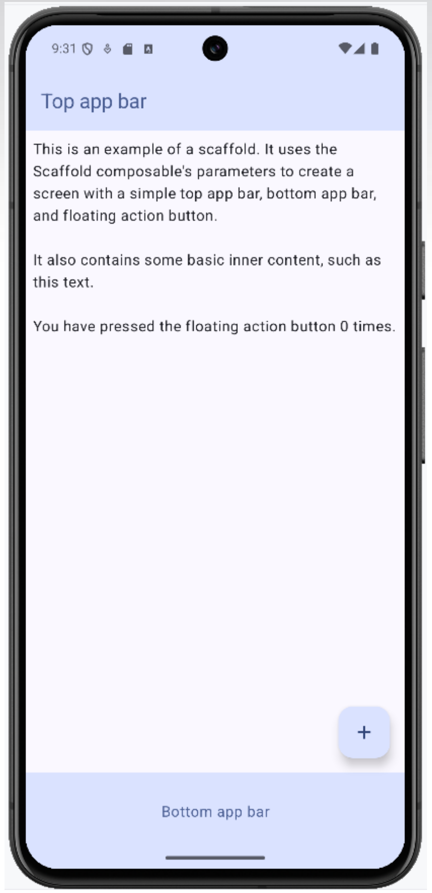
You have pressed the floating action button 0 times.

Bottom app bar

Scaffold

Scaffold Components

```
Scaffold(    topBar = { /* TopAppBar goes here */ },
              bottomBar = { /* NavigationBar goes here */ },
              floatingActionButton = { /* FAB goes here */ }
        ) { innerPadding ->
              // Content goes here (Apply innerPadding)}
```



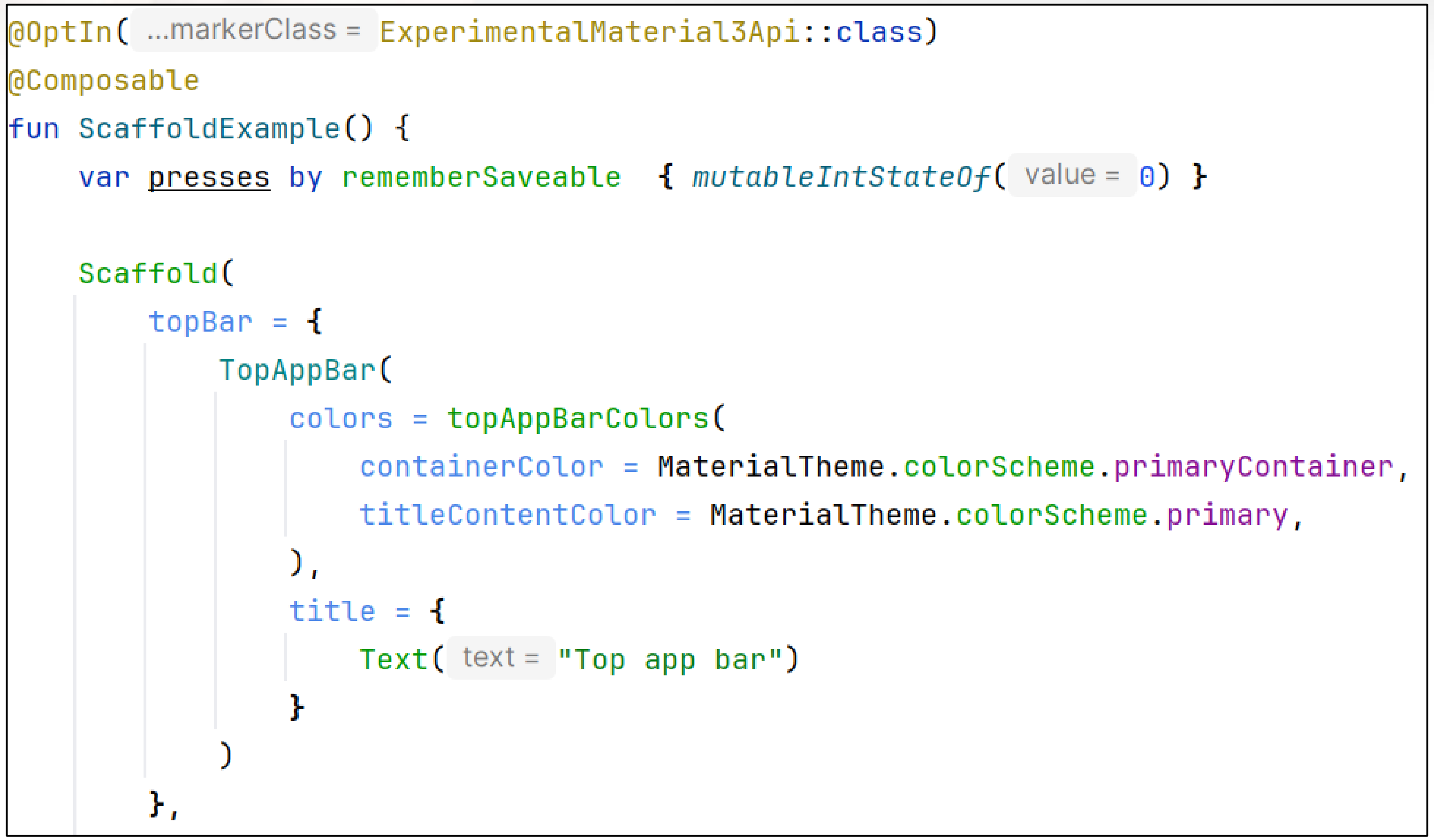
This is an example of a scaffold. It uses the Scaffold composable's parameters to create a screen with a simple top app bar, bottom app bar, and floating action button.

It also contains some basic inner content, such as this text.

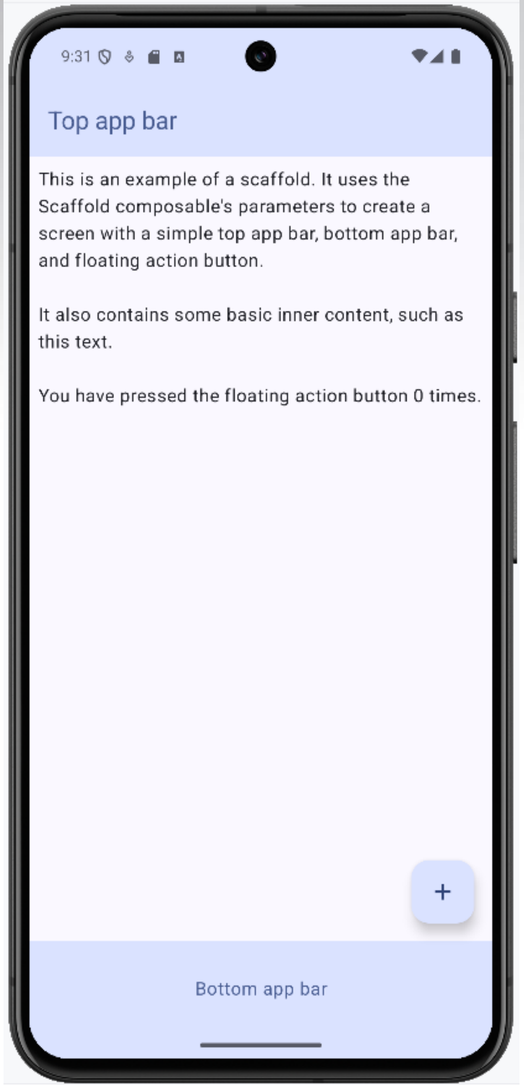
You have pressed the floating action button 0 times.

Scaffold

Scaffold Example



```
@OptIn( ...markerClass = ExperimentalMaterial3Api::class)
@Composable
fun ScaffoldExample() {
    var presses by rememberSaveable  { mutableIntStateOf( value = 0) }

    Scaffold(
        topBar = {
            TopAppBar(
                colors = topAppBarColors(
                    containerColor = MaterialTheme.colorScheme.primaryContainer,
                    titleContentColor = MaterialTheme.colorScheme.primary,
                ),
                title = {
                    Text( text = "Top app bar")
                }
            )
        },
```



Top app bar

This is an example of a scaffold. It uses the Scaffold composable's parameters to create a screen with a simple top app bar, bottom app bar, and floating action button.

It also contains some basic inner content, such as this text.

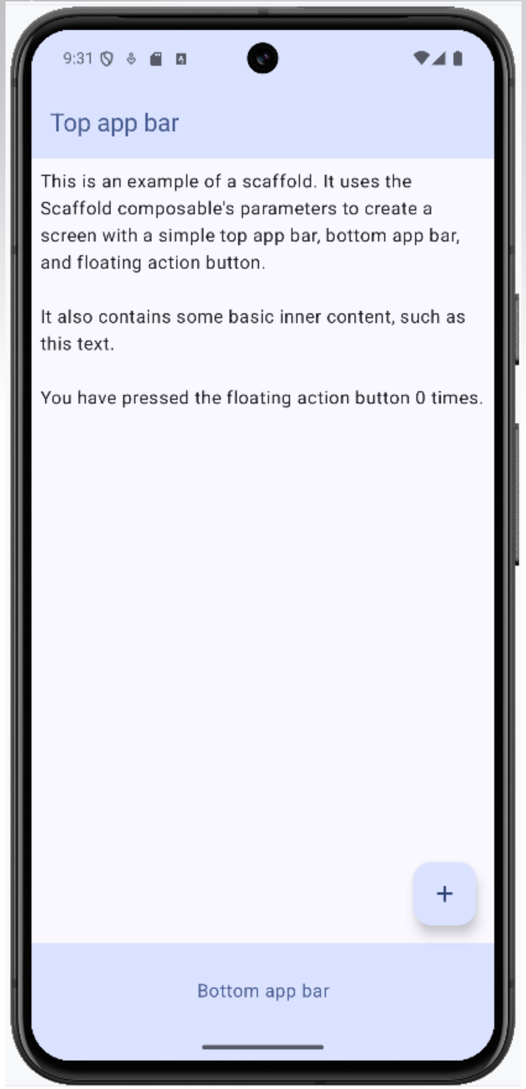
You have pressed the floating action button 0 times.

https://developer.android.com/develop/ui/compose/quick-guides/content/create-scaffold

Scaffold

Scaffold Example (Cont.)

```
bottomBar = {
    BottomAppBar(
        containerColor = MaterialTheme.colorScheme.primaryContainer,
        contentColor = MaterialTheme.colorScheme.primary,
    ) {
        Text(
            modifier = Modifier
                .fillMaxWidth(),
            textAlign = TextAlign.Center,
            text = "Bottom app bar",
        )
    }
},
floatingActionButton = {
    FloatingActionButton(onClick = { presses++ }) {
        Icon( imageVector = Icons.Default.Add, contentDescription = "Add")
    }
}
```


9:31

Top app bar

This is an example of a scaffold. It uses the Scaffold composable's parameters to create a screen with a simple top app bar, bottom app bar, and floating action button.

It also contains some basic inner content, such as this text.

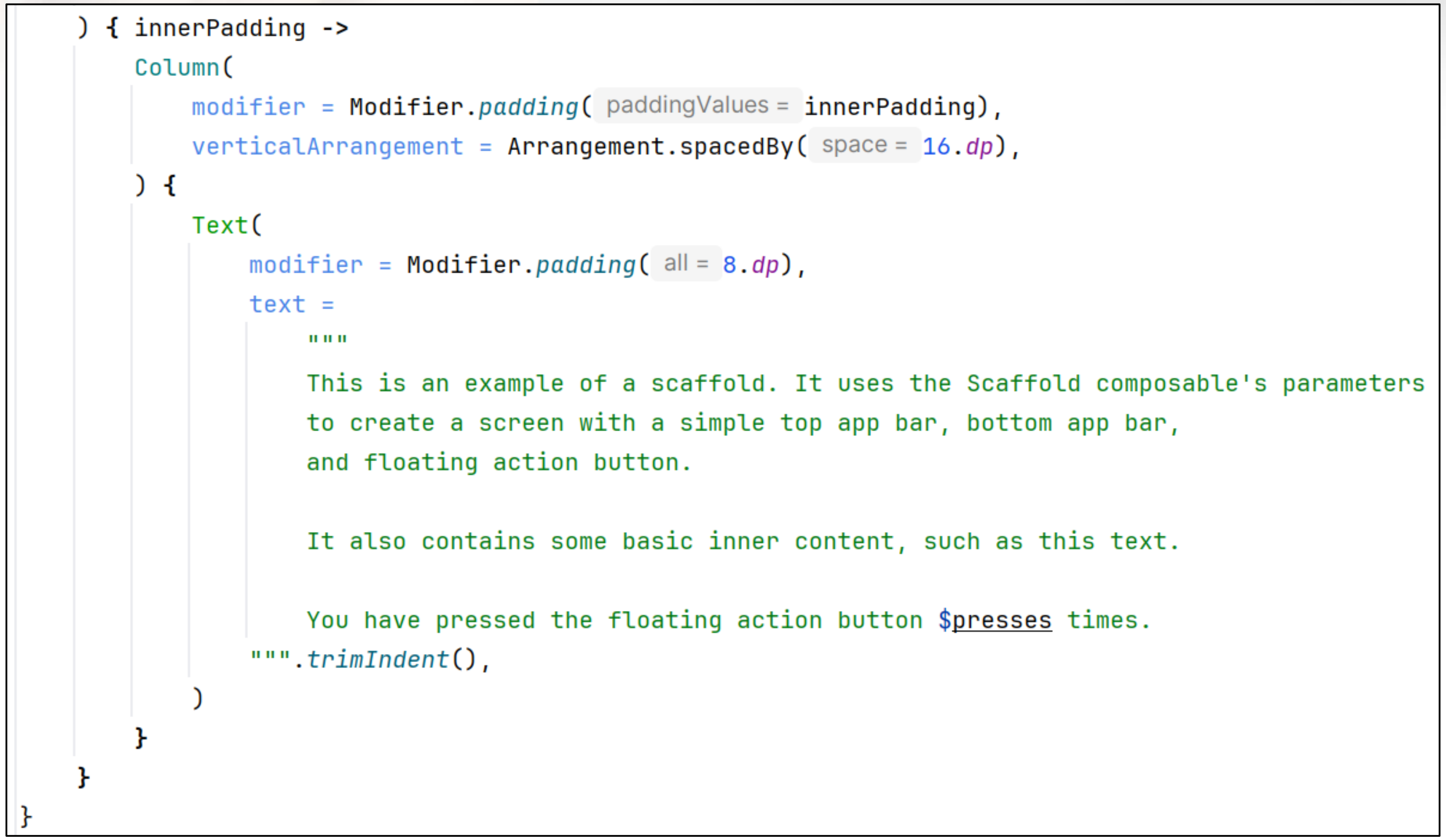
You have pressed the floating action button 0 times.

Bottom app bar

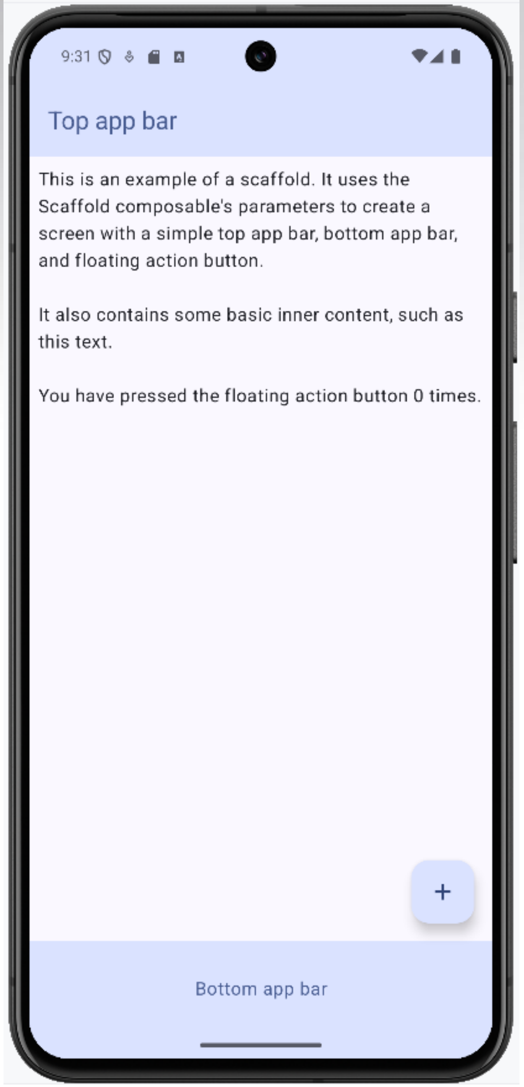
https://developer.android.com/develop/ui/compose/quick-guides/content/create-scaffold

Scaffold

Scaffold Example (Cont.)



```
    ) { innerPadding ->
        Column(
            modifier = Modifier.padding( paddingValues = innerPadding),
            verticalArrangement = Arrangement.spacedBy( space = 16.dp),
        ) {
            Text(
                modifier = Modifier.padding( all = 8.dp),
                text =
                    """
                    This is an example of a scaffold. It uses the Scaffold composable's parameters
                    to create a screen with a simple top app bar, bottom app bar,
                    and floating action button.

                    It also contains some basic inner content, such as this text.

                    You have pressed the floating action button $presses times.
                """.trimIndent(),
            )
        }
    }
}
```



9:31

Top app bar

This is an example of a scaffold. It uses the Scaffold composable's parameters to create a screen with a simple top app bar, bottom app bar, and floating action button.

It also contains some basic inner content, such as this text.

You have pressed the floating action button 0 times.

Bottom app bar

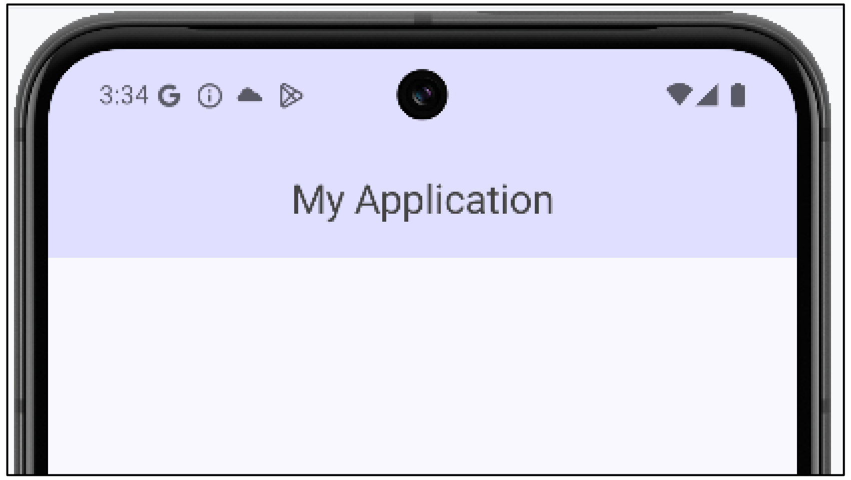
https://developer.android.com/develop/ui/compose/quick-guides/content/create-scaffold

Top App Bar Navigation

Top app bars display information and actions related to the current screen. They are displayed at the top of the screen.

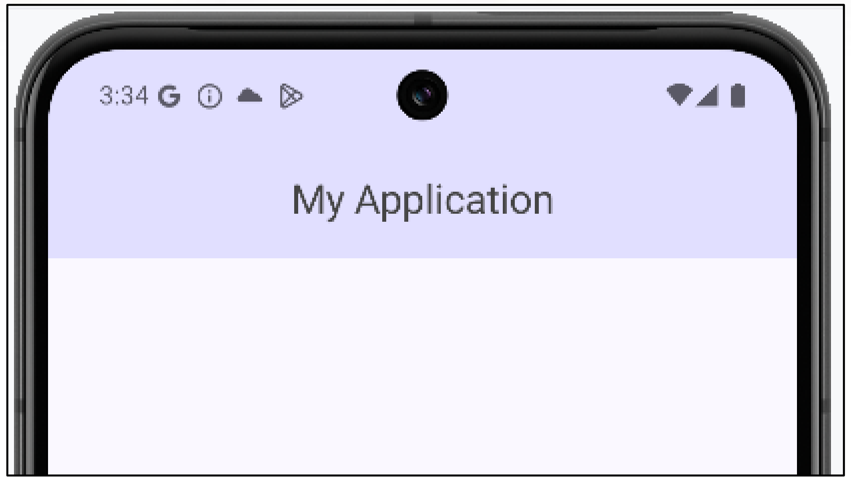
There are four types of TopAppbars design system (Material 3):

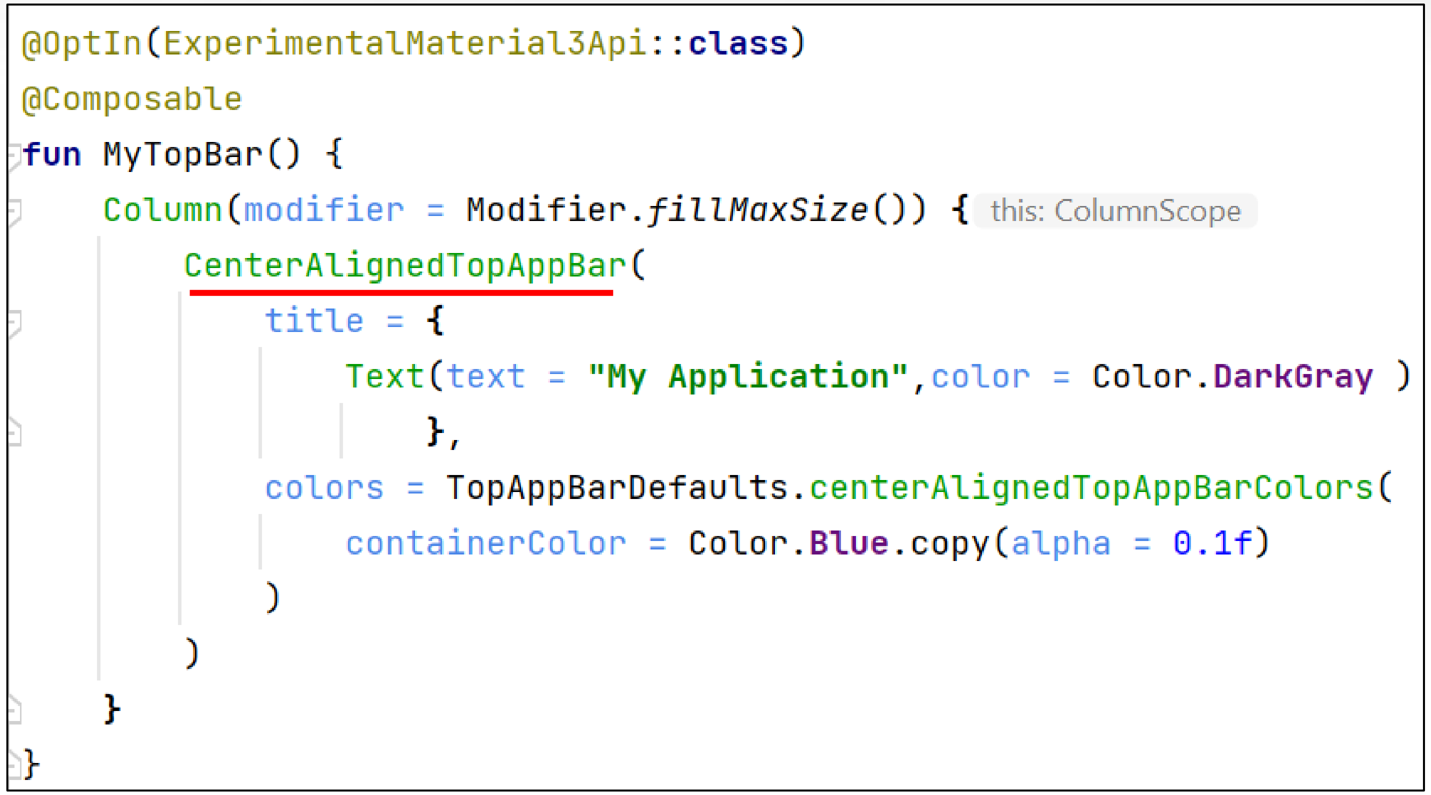
1. Center-aligned
2. Small
3. Medium
4. Large



Top App Bar Navigation

1. Center-aligned TopAppBar: CenterAlignedTopAppBar()

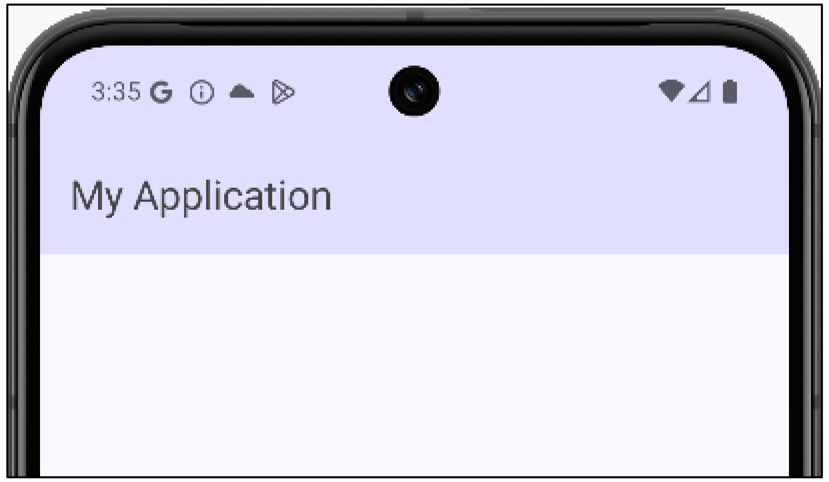


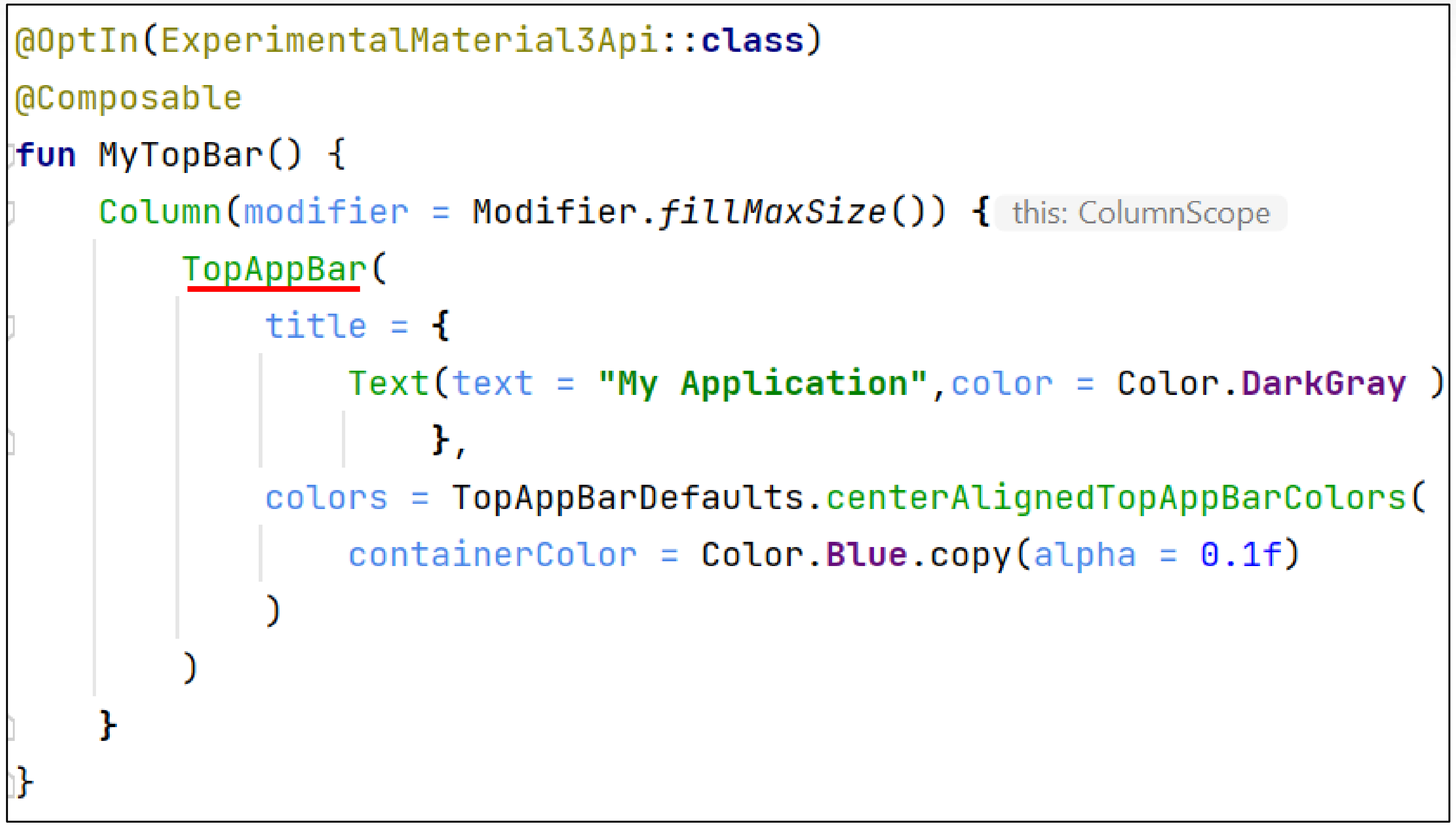


```
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun MyTopBar() {
    Column(modifier = Modifier.fillMaxSize()) { this: ColumnScope
        CenterAlignedTopAppBar(
            title = {
                Text(text = "My Application",color = Color.DarkGray )
                    },
            colors = TopAppBarDefaults.centerAlignedTopAppBarColors(
                containerColor = Color.Blue.copy(alpha = 0.1f)
            )
        )
    }
}
```

Top App Bar Navigation

2. Small TopAppBar: TopAppBar()

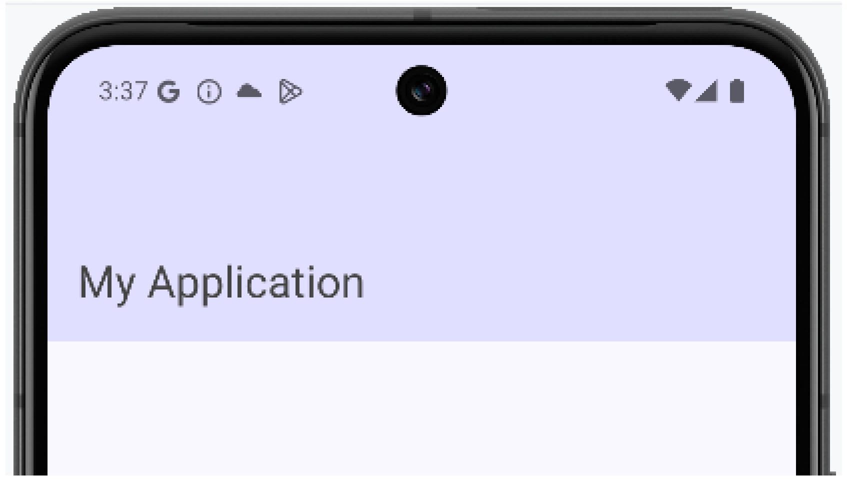


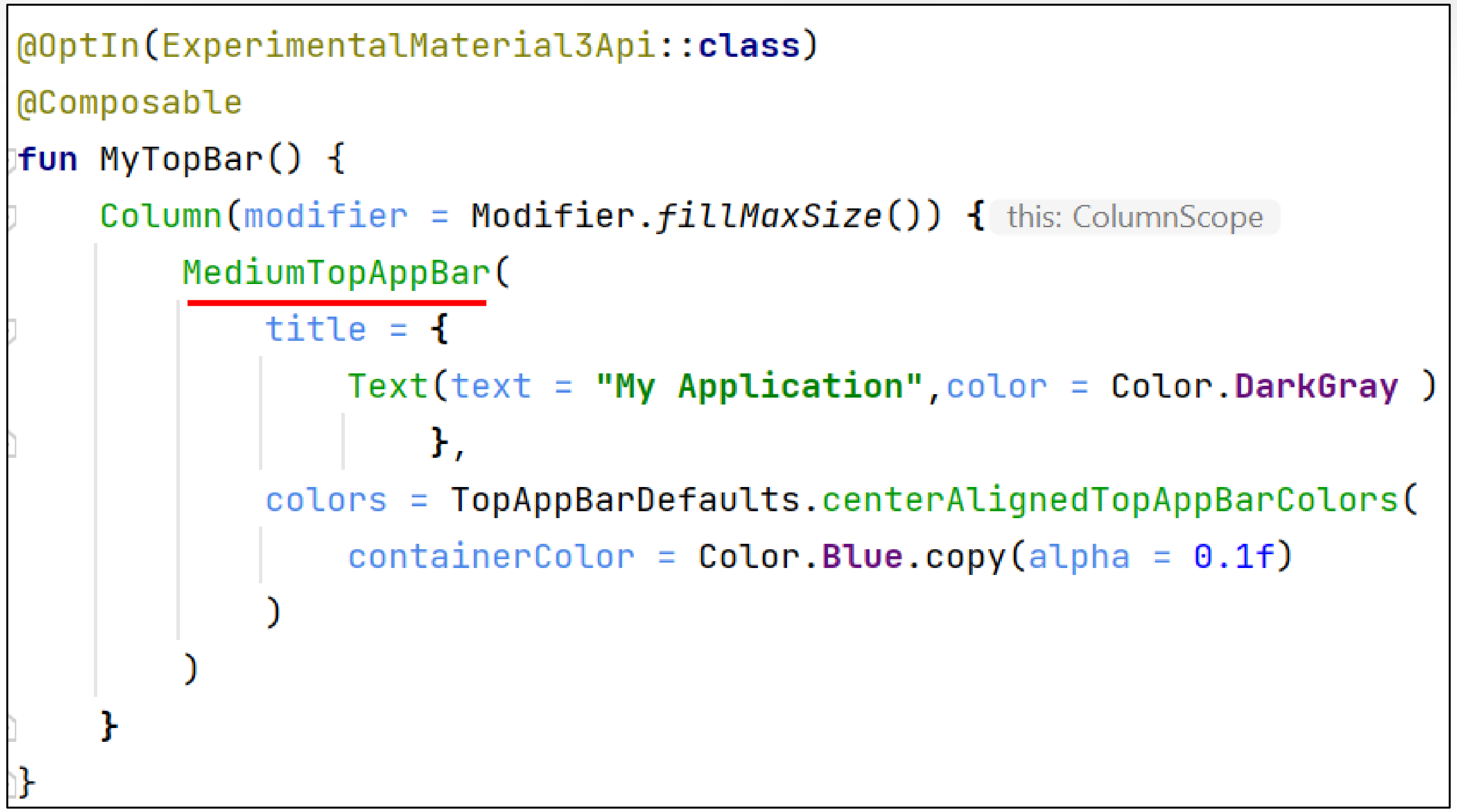


```
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun MyTopBar() {
    Column(modifier = Modifier.fillMaxSize()) { this: ColumnScope
        TopAppBar(
            title = {
                Text(text = "My Application",color = Color.DarkGray )
                    },
            colors = TopAppBarDefaults.centerAlignedTopAppBarColors(
                containerColor = Color.Blue.copy(alpha = 0.1f)
            )
        )
    }
}
```

Top App Bar Navigation

3. Medium TopAppBar: MediumTopAppBar()

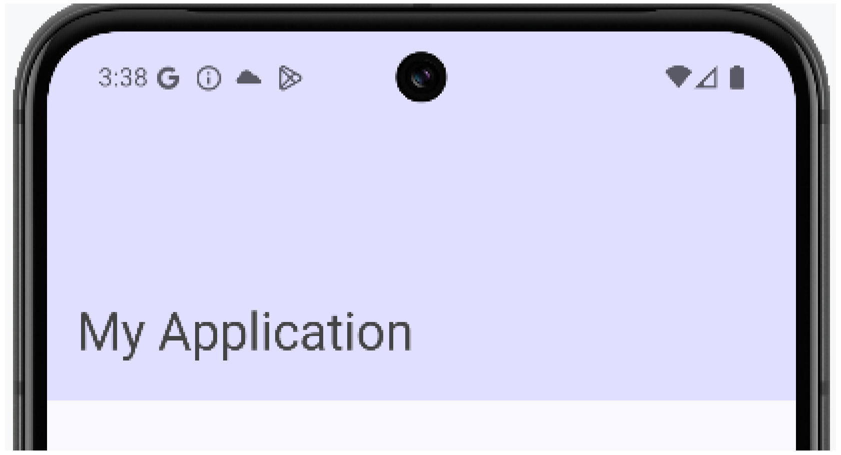


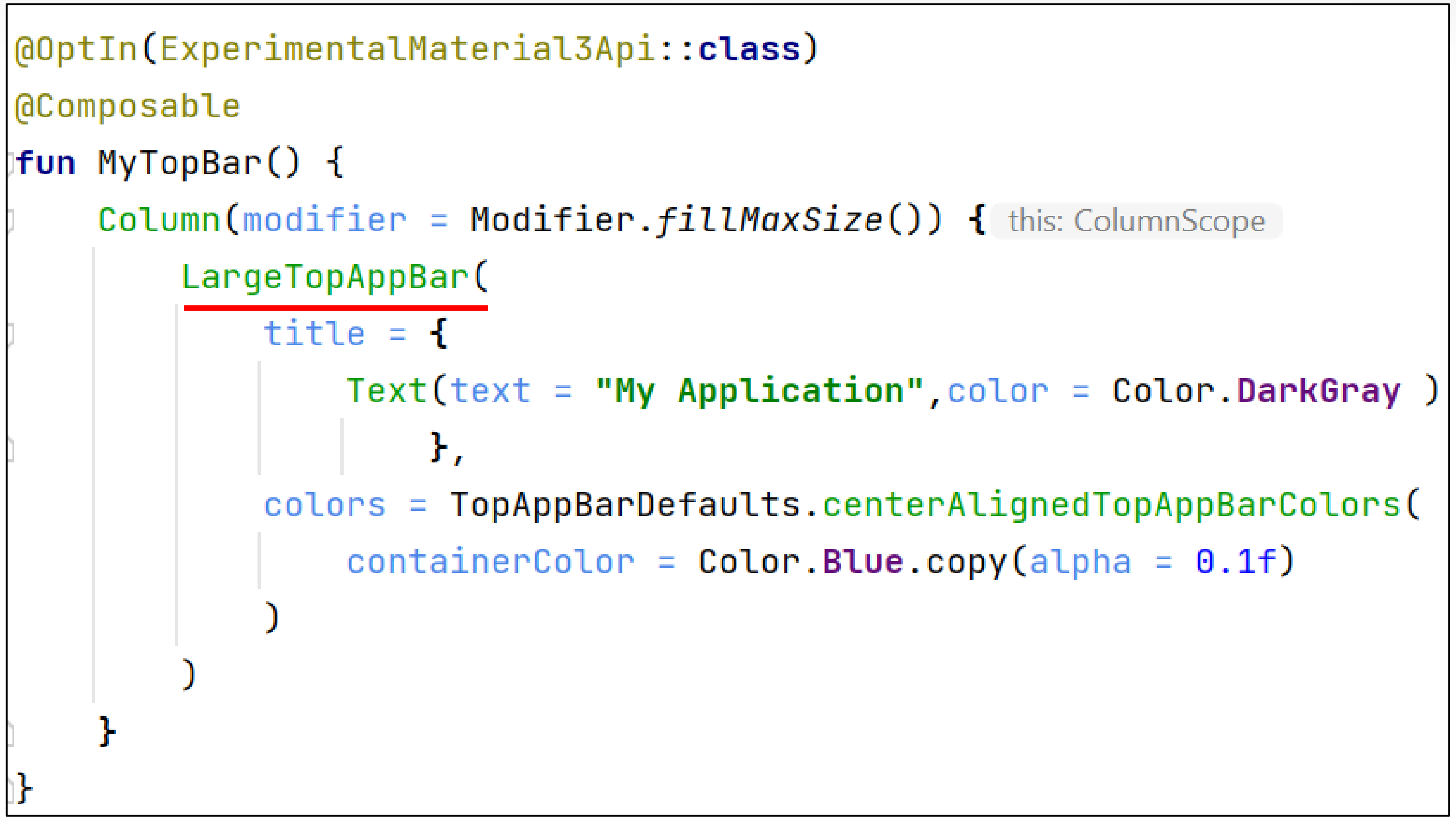


```
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun MyTopBar() {
    Column(modifier = Modifier.fillMaxSize()) { this: ColumnScope
        MediumTopAppBar(
            title = {
                Text(text = "My Application",color = Color.DarkGray )
                    },
            colors = TopAppBarDefaults.centerAlignedTopAppBarColors(
                containerColor = Color.Blue.copy(alpha = 0.1f)
            )
        )
    }
}
```

Top App Bar Navigation

4. Large TopAppBar: LargeTopAppBar()



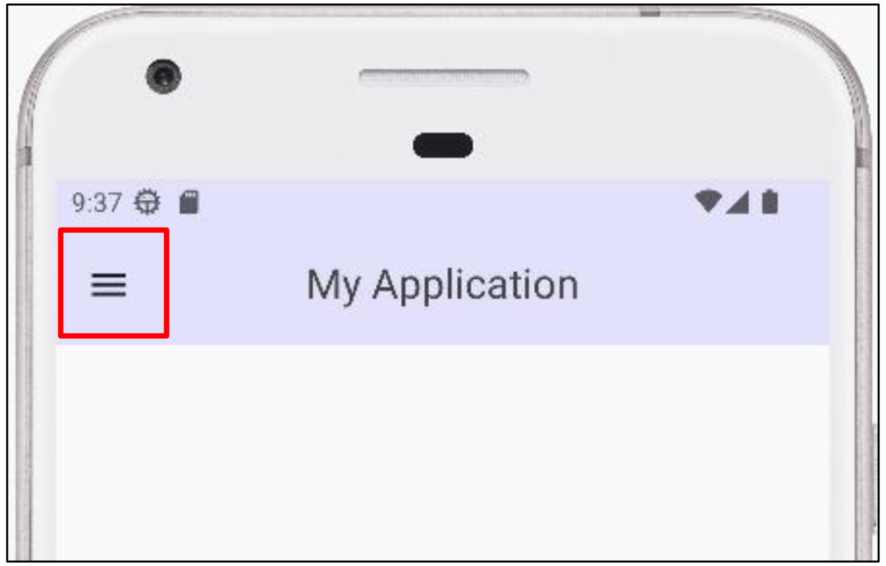


```
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun MyTopBar() {
    Column(modifier = Modifier.fillMaxSize()) { this: ColumnScope
        LargeTopAppBar(
            title = {
                Text(text = "My Application",color = Color.DarkGray )
                    },
            colors = TopAppBarDefaults.centerAlignedTopAppBarColors(
                containerColor = Color.Blue.copy(alpha = 0.1f)
            )
        )
    }
}
```

Top App Bar Navigation

1. Navigation Icon

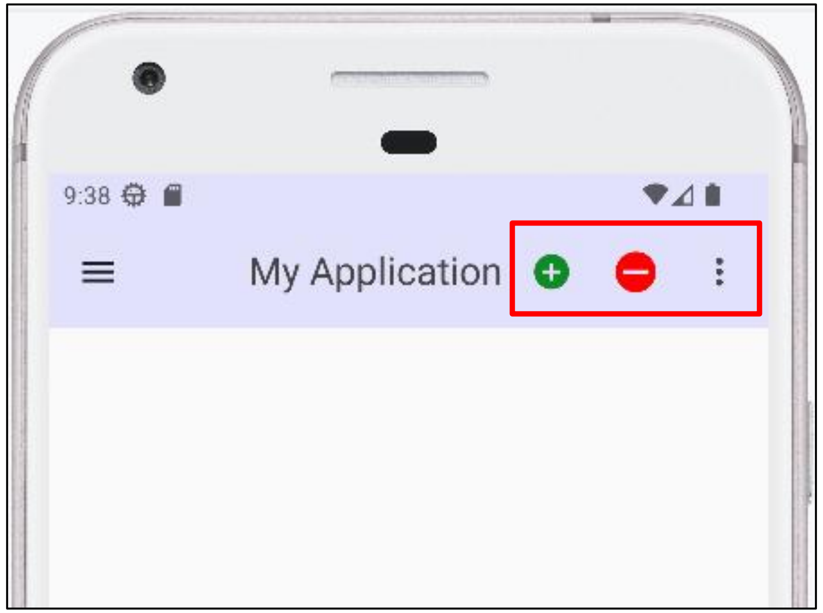
We can set the navigation icon using the navigationIcon parameter. Use the IconButton API to add the icon.



Top App Bar Navigation

2. Menu Icons or Actions:

The actions parameter is used to add the icons at the end of the top app bar. It is a row scope. So, if you add multiple items, they are placed horizontally.



Top App Bar Navigation

Navigation Icon: Navigation Drawer

A menu that slides in from the left. It shows different screens (destinations) to the user. We can also open it by swiping from the left.

Jetpack Compose provides two types of Navigation Drawers:

1. Modal Navigation Drawer
2. Dismissible Navigation Drawer

Top App Bar Navigation

Navigation Icon: Navigation Drawer

Class and function to create navigation drawer:

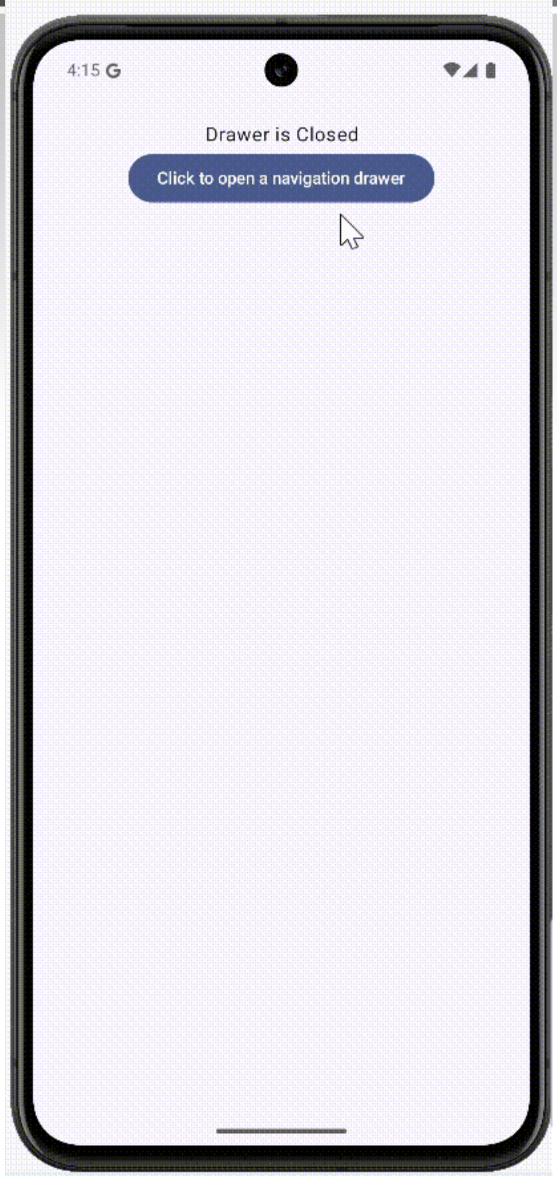
- NavigationDrawerData data class is used to hold the drawer item’s name and icon.
- prepareNavigationDrawerItems() is created the list and adding the drawer items.
- drawerContent parameter to add the items.
- ModalDrawerSheet to setup within the sheet
- NavigationDrawerItem() method to display the items

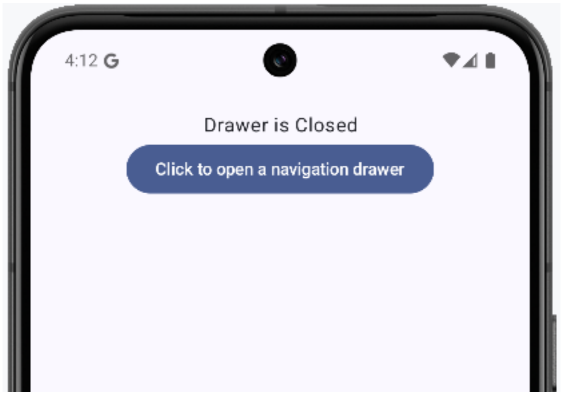
Top App Bar Navigation

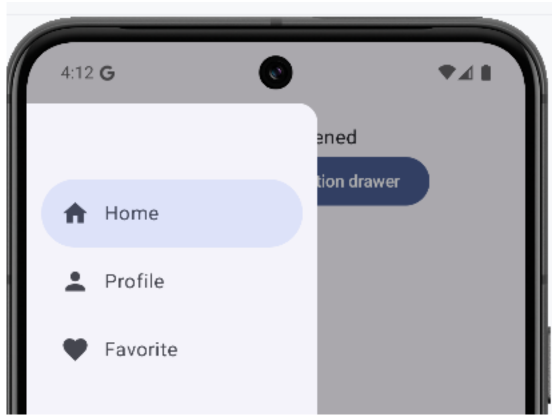
Navigation Icon: Navigation Drawer

1. Modal Navigation Drawer:

   When it is opened, users cannot interact with the rest of the app's content. It shows a semi-transparent overlay (scrim) on the content

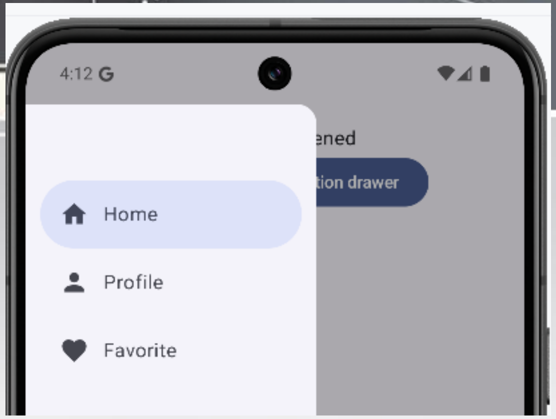






Top App Bar Navigation

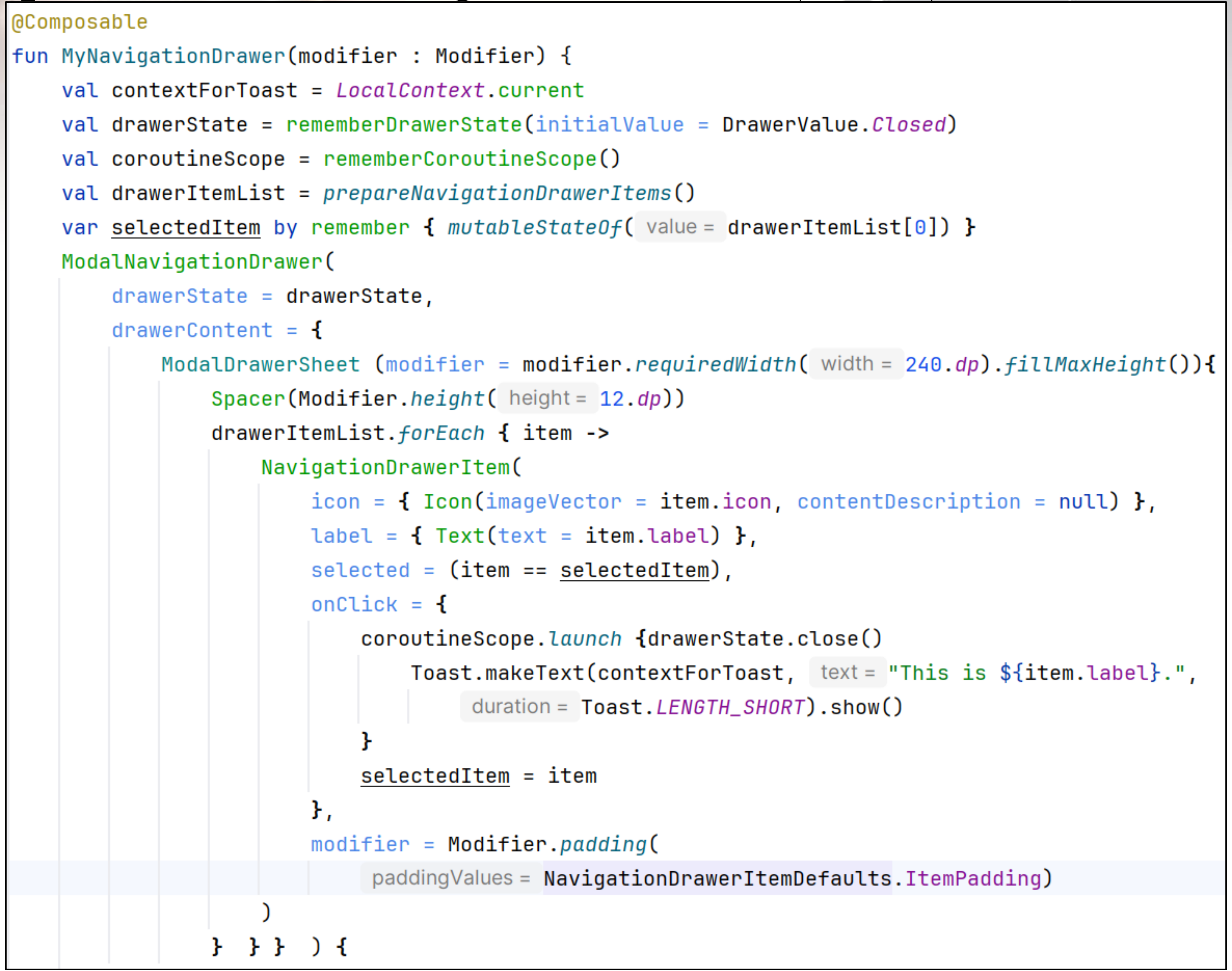
Example of Modal Navigation Drawer:



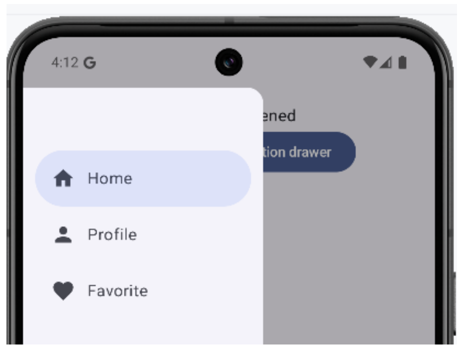
```
data class NavigationDrawerData(val label: String, val icon: ImageVector)


1 Usage
private fun prepareNavigationDrawerItems(): List<NavigationDrawerData> {
    val drawerItemsList = arrayListOf<NavigationDrawerData>()
    // add items
    drawerItemsList.add(NavigationDrawerData(label = "Home", icon = Icons.Filled.Home))
    drawerItemsList.add(NavigationDrawerData(label = "Profile", icon = Icons.Filled.Person))
    drawerItemsList.add(NavigationDrawerData(label = "Favorite", icon = Icons.Filled.Favorite))
    return drawerItemsList
}
```

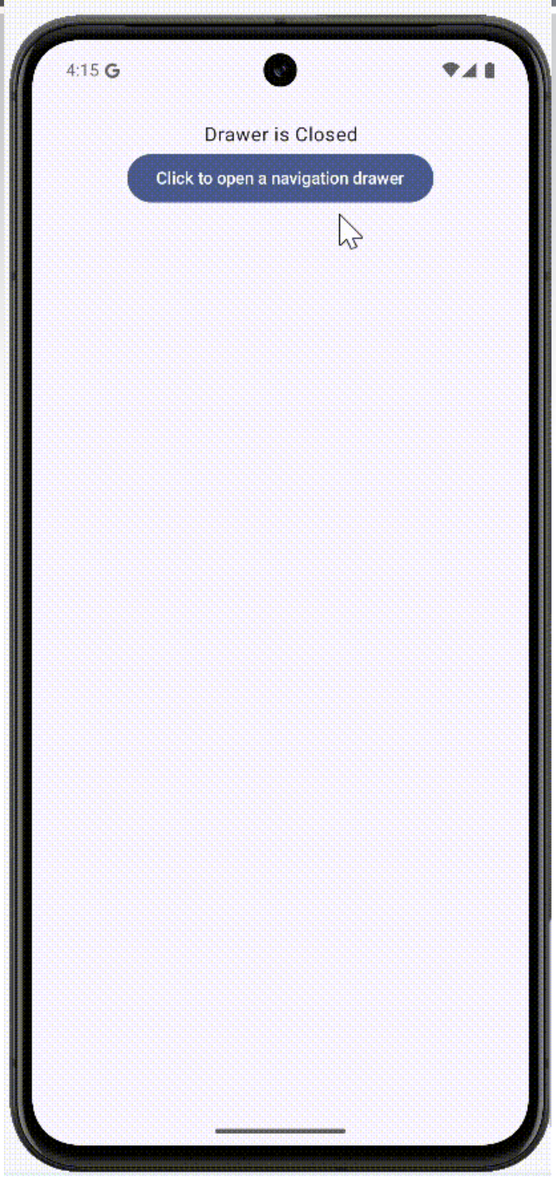
Top App Bar Navigation

Example of Modal Navigation Drawer(Cont.):



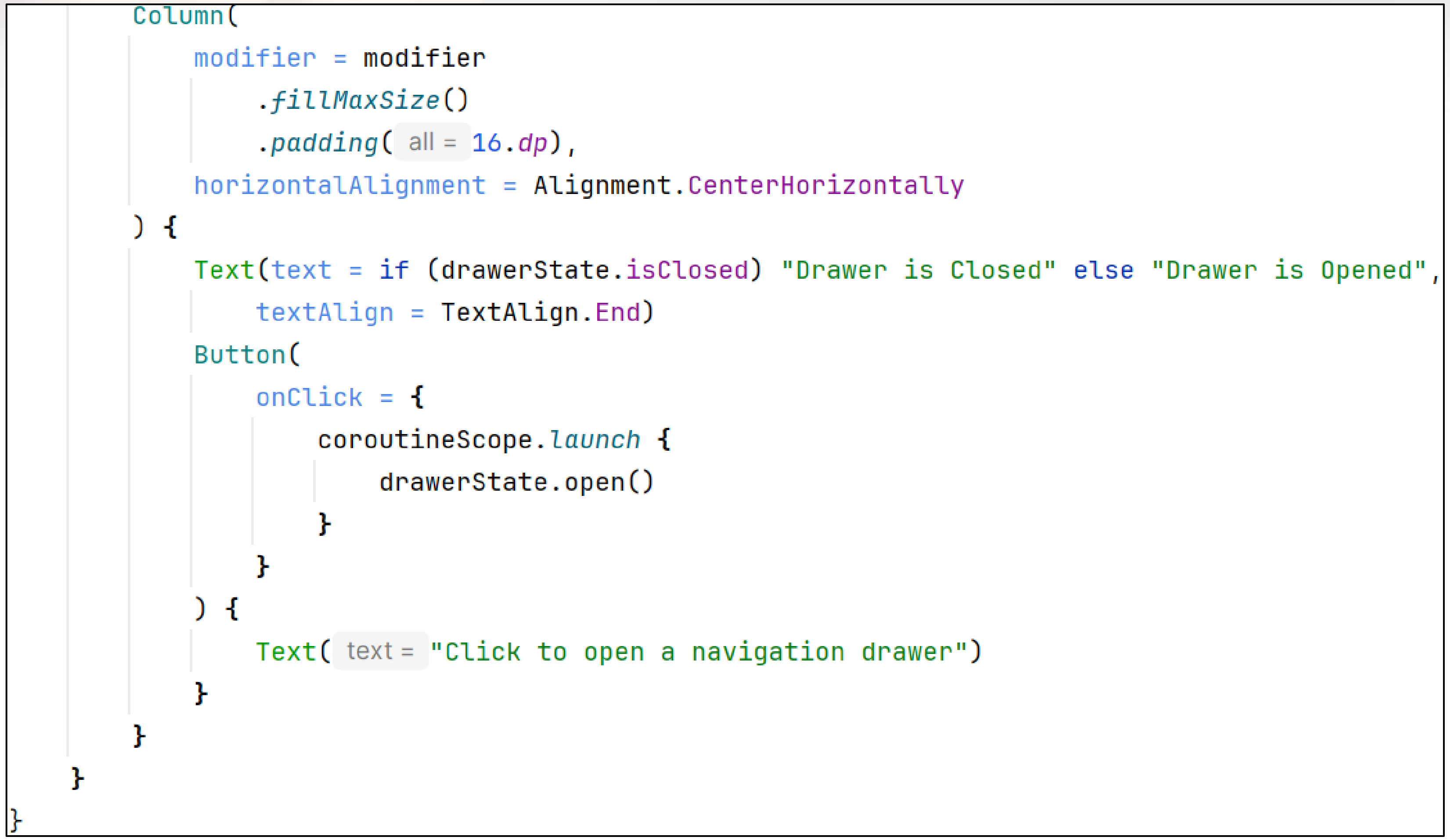
```kotlin
@Composable
fun MyNavigationDrawer(modifier : Modifier) {
    val contextForToast = LocalContext.current
    val drawerState = rememberDrawerState(initialValue = DrawerValue.Closed)
    val coroutineScope = rememberCoroutineScope()
    val drawerItemList = prepareNavigationDrawerItems()
    var selectedItem by remember { mutableStateOf( value = drawerItemList[0]) }
    ModalNavigationDrawer(
        drawerState = drawerState,
        drawerContent = {
            ModalDrawerSheet (modifier = modifier.requiredWidth( width = 240.dp).fillMaxHeight()){
                Spacer(Modifier.height( height = 12.dp))
                drawerItemList.forEach { item ->
                    NavigationDrawerItem(
                        icon = { Icon(imageVector = item.icon, contentDescription = null) },
                        label = { Text(text = item.label) },
                        selected = (item == selectedItem),
                        onClick = {
                            coroutineScope.launch {drawerState.close()
                                Toast.makeText(contextForToast,  text = "This is ${item.label}.",
                                     duration = Toast.LENGTH_SHORT).show()
                            }
                            selectedItem = item
                        },
                        modifier = Modifier.padding(
                            paddingValues = NavigationDrawerItemDefaults.ItemPadding)
                    )
            }  } }  ) {
```



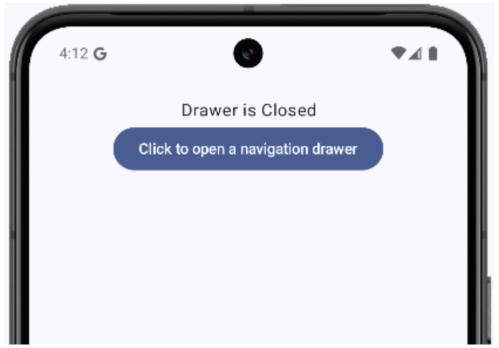


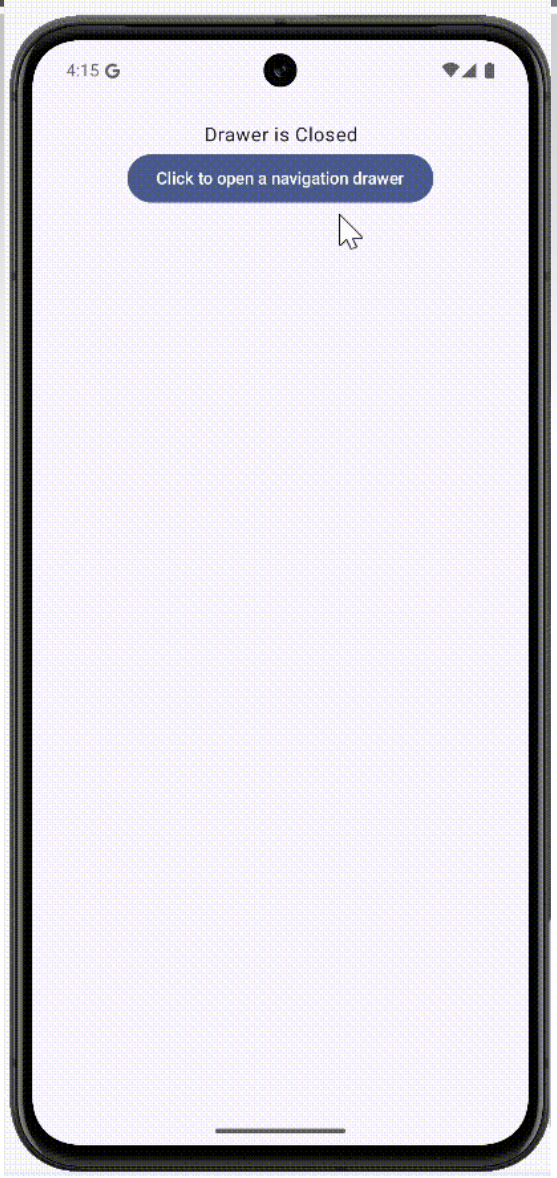
Top App Bar Navigation

Example of Modal Navigation Drawer(Cont.):



```
        Column(
            modifier = modifier
                .fillMaxSize()
                .padding( all = 16.dp),
            horizontalAlignment = Alignment.CenterHorizontally
        ) {
            Text(text = if (drawerState.isClosed) "Drawer is Closed" else "Drawer is Opened",
                textAlign = TextAlign.End)
            Button(
                onClick = {
                    coroutineScope.launch {
                        drawerState.open()
                    }
                }
            ) {
                Text( text = "Click to open a navigation drawer")
            }
        }
    }
}
```





Top App Bar Navigation

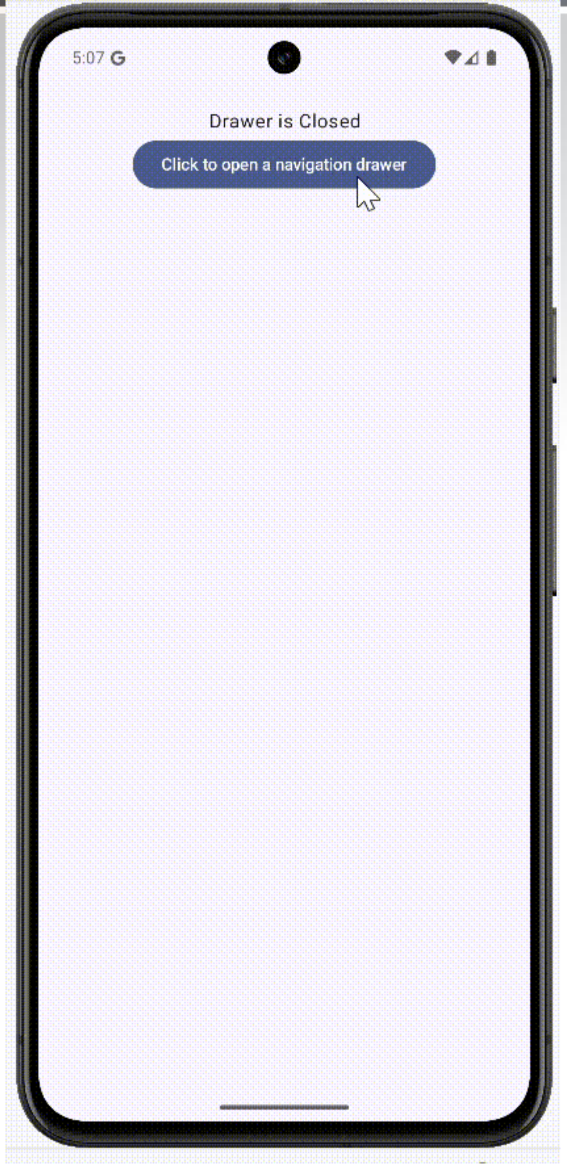
Navigation Icon: Navigation Drawer

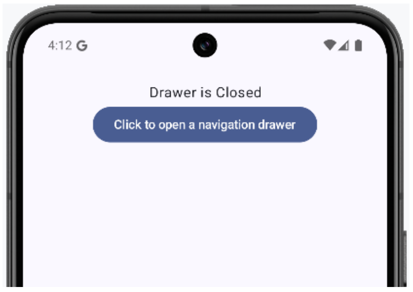
2. Dismissible Navigation Drawer:

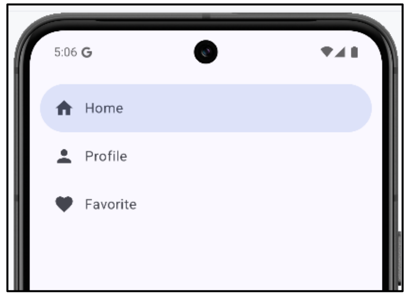
It is similar to the modal navigation drawer, but when it is opened, it covers the whole screen.

**The code is similar to the Modal Navigation Drawer.

- Replaced ModalNavigationDrawer() with DismissibleNavigationDrawer()
- Replaced ModalDrawerSheet with DismissibleDrawerSheet.
- DismissibleDrawerSheet edits modifier to modifier =modifier.*fillMaxSize*()

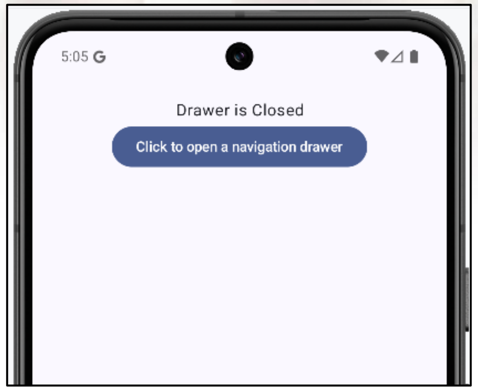


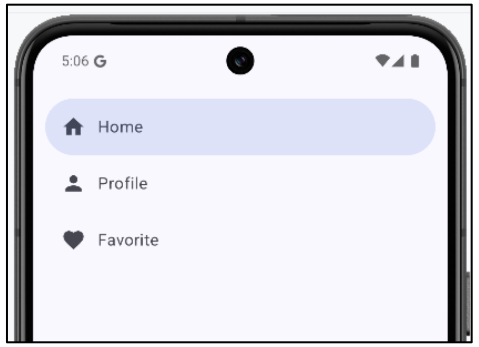


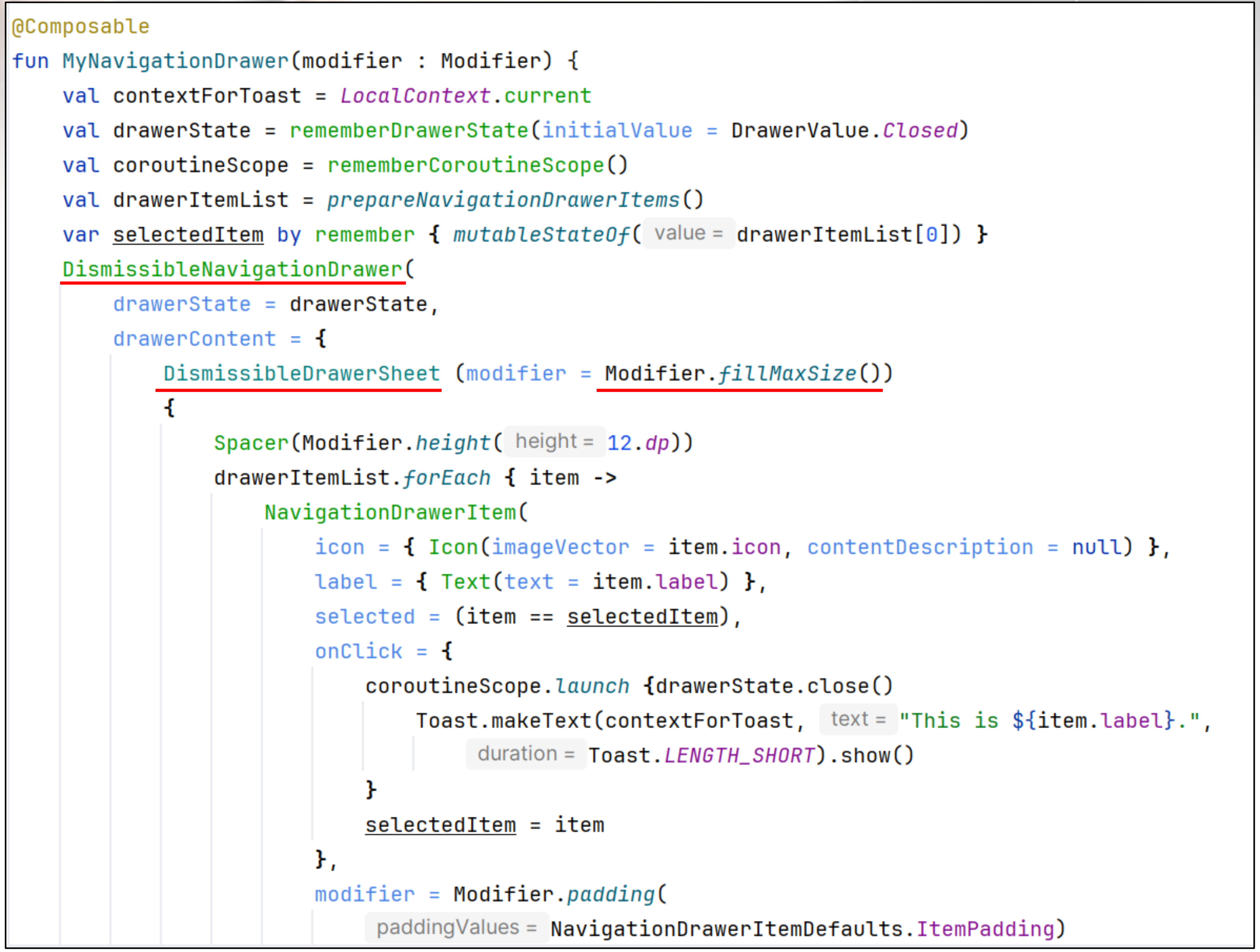


Top App Bar Navigation

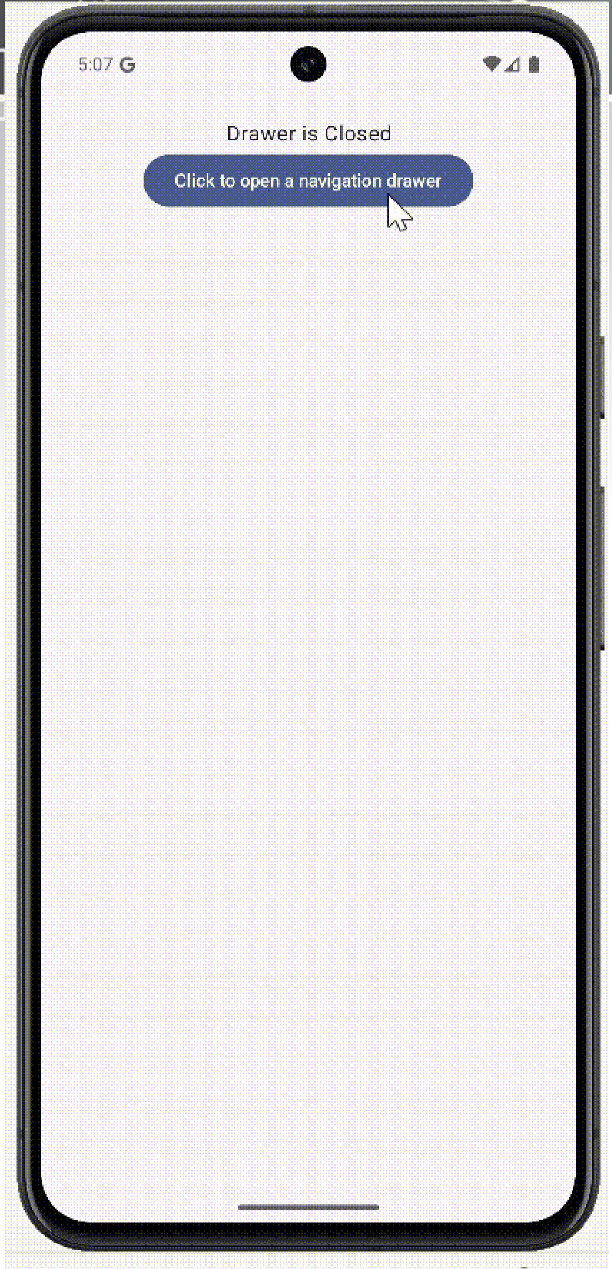
Example of Dismissible Navigation Drawer:





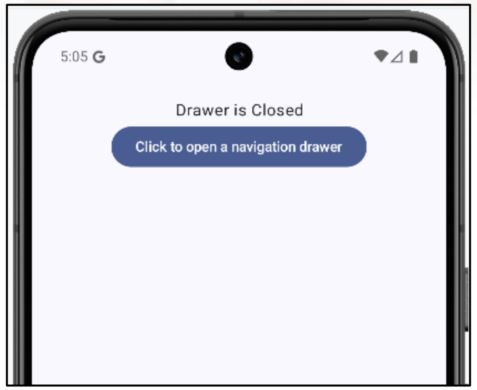


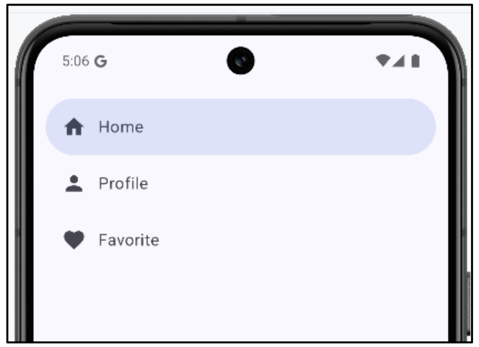
```
@Composable
fun MyNavigationDrawer(modifier : Modifier) {
    val contextForToast = LocalContext.current
    val drawerState = rememberDrawerState(initialValue = DrawerValue.Closed)
    val coroutineScope = rememberCoroutineScope()
    val drawerItemList = prepareNavigationDrawerItems()
    var selectedItem by remember { mutableStateOf( value = drawerItemList[0]) }
    DismissibleNavigationDrawer(
        drawerState = drawerState,
        drawerContent = {
            DismissibleDrawerSheet (modifier = Modifier.fillMaxSize())
            {
                Spacer(Modifier.height( height = 12.dp))
                drawerItemList.forEach { item ->
                    NavigationDrawerItem(
                        icon = { Icon(imageVector = item.icon, contentDescription = null) },
                        label = { Text(text = item.label) },
                        selected = (item == selectedItem),
                        onClick = {
                            coroutineScope.launch {drawerState.close()
                                Toast.makeText(contextForToast,  text = "This is ${item.label}.",
                                     duration = Toast.LENGTH_SHORT).show()
                            }
                            selectedItem = item
                        },
                        modifier = Modifier.padding(
                            paddingValues = NavigationDrawerItemDefaults.ItemPadding)
```



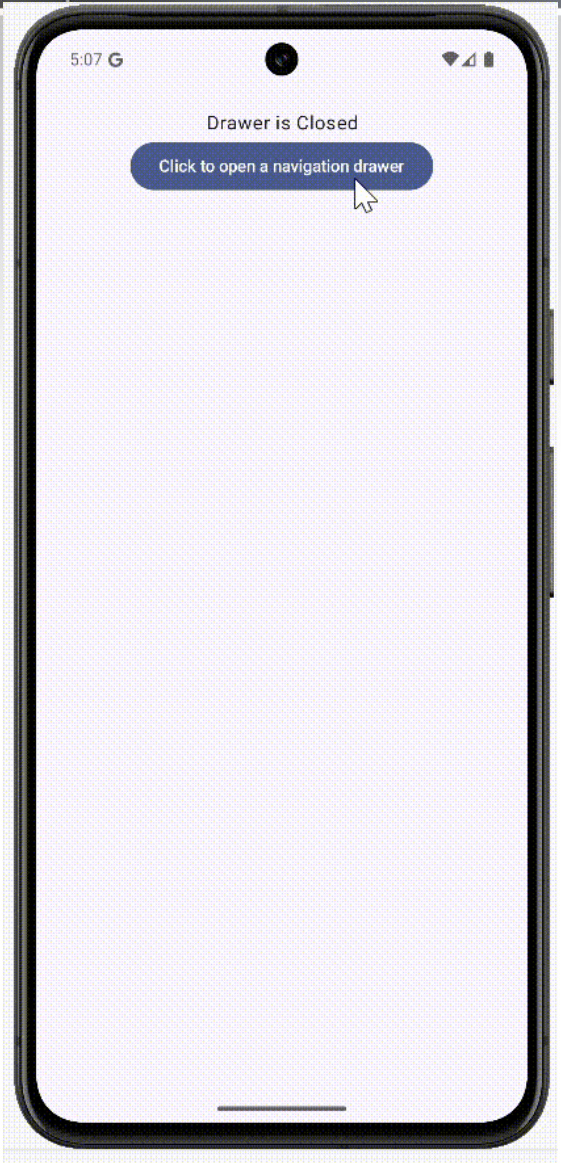
Top App Bar Navigation

Example of Dismissible Navigation Drawer(Cont.):





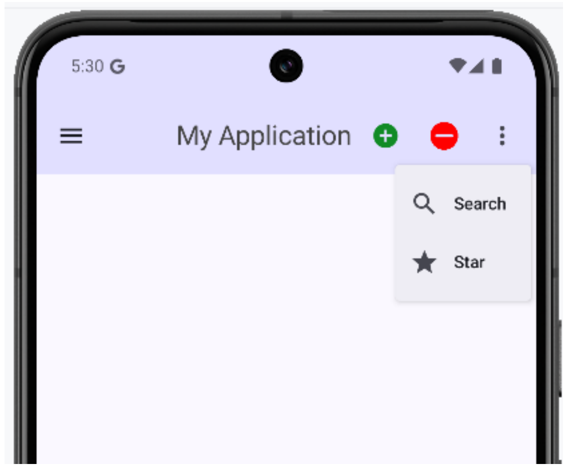
```
        Column(
            modifier = modifier
                .fillMaxSize()
                .padding( all = 16.dp),
            horizontalAlignment = Alignment.CenterHorizontally
        ) {
            Text(text = if (drawerState.isClosed) "Drawer is Closed" else "Drawer is Opened",
                textAlign = TextAlign.End)
            Button(
                onClick = {
                    coroutineScope.launch {
                        drawerState.open()
                    }
                }
            ) {
                Text( text = "Click to open a navigation drawer")
            }
        }
    }
}
```



Menu

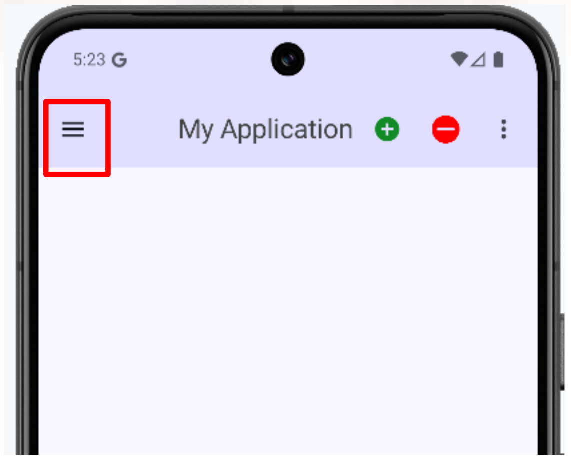
A Menu is a common user interface component in many types of applications.

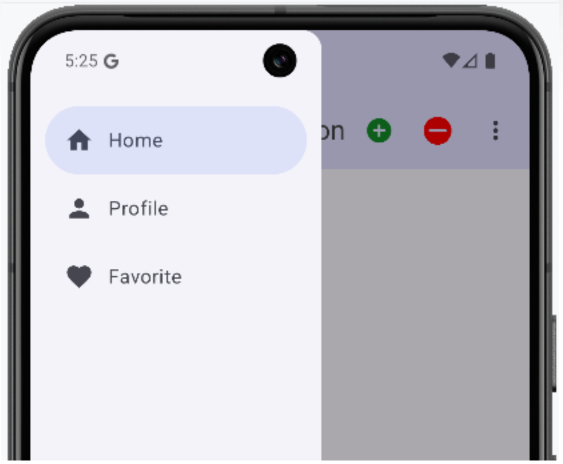
- Overflow Menu: Hidden menu button revealing less frequently used actions, typically triggered by three dots.
- Dropdown Menu: Cascading menu triggered by a button, displaying options related to the current context.

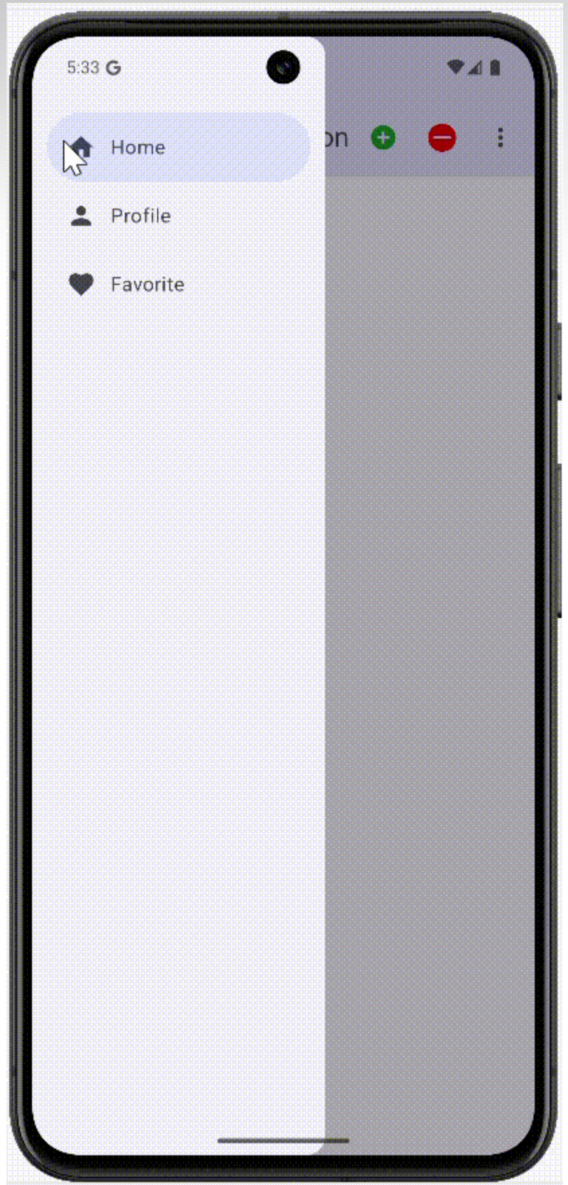


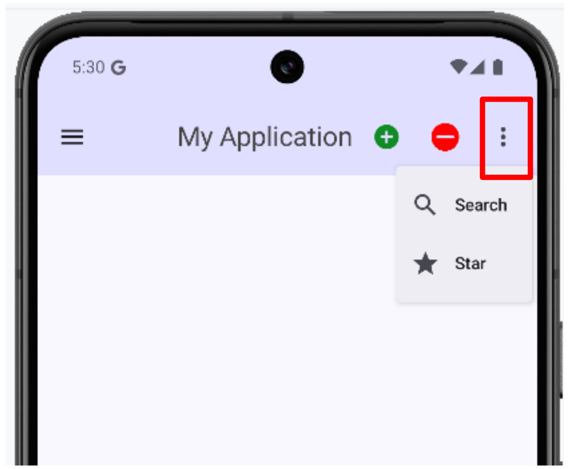
Top App Bar Navigation

Example of Top App Bar Navigation:





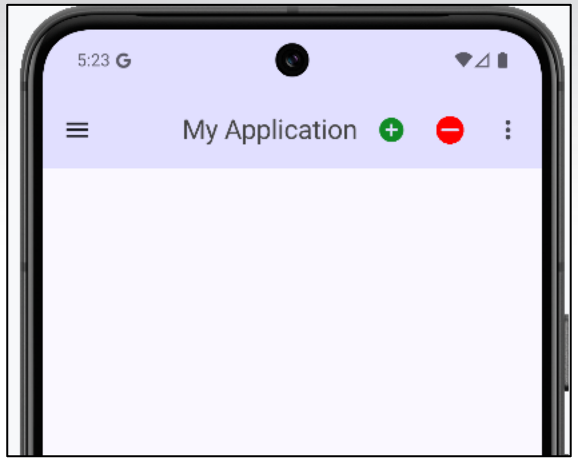


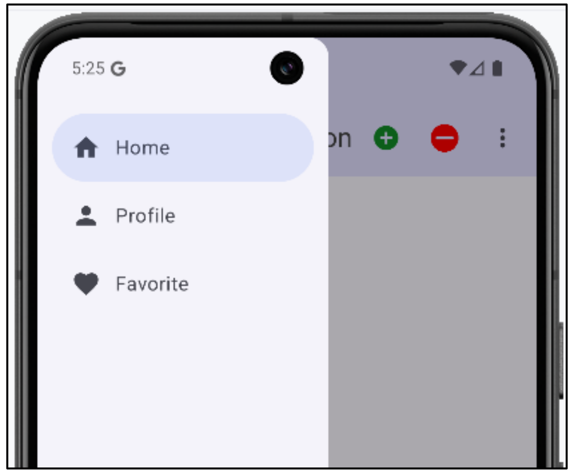


Top App Bar Navigation

Example of Top App Bar Navigation:

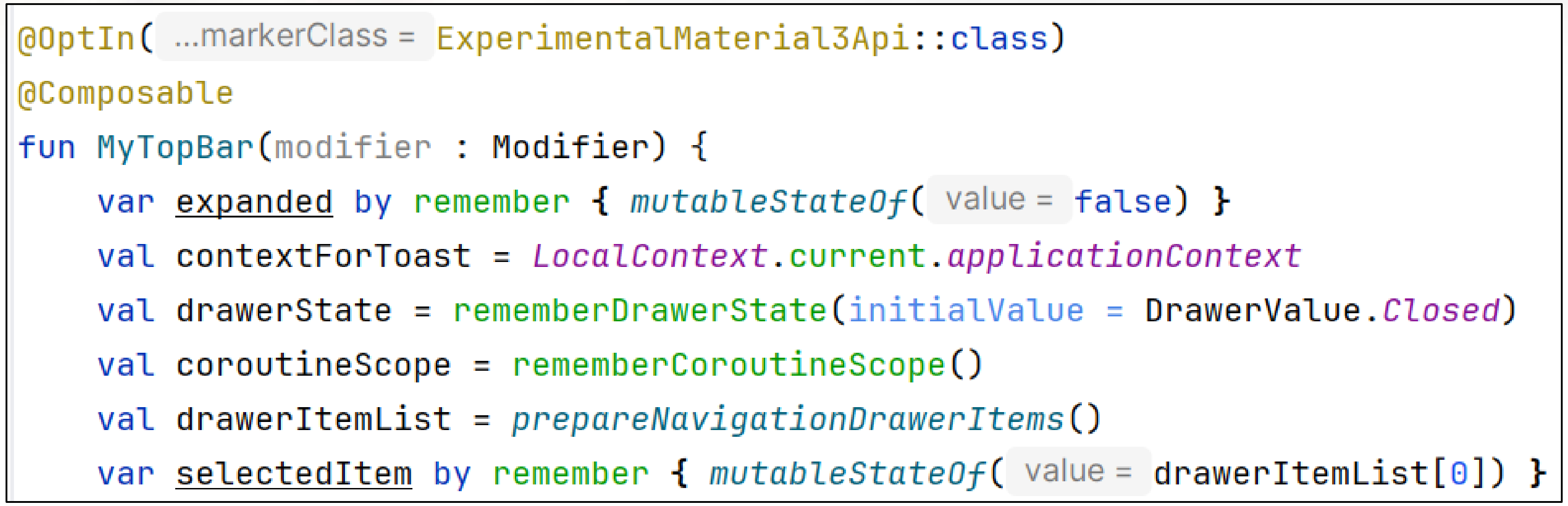
```
data class NavigationDrawerData(val label: String, val icon: ImageVector)

1 Usage
private fun prepareNavigationDrawerItems(): List<NavigationDrawerData> {
    val drawerItemsList = arrayListOf<NavigationDrawerData>()
    // add items
    drawerItemsList.add(NavigationDrawerData(label = "Home", icon = Icons.Filled.Home))
    drawerItemsList.add(NavigationDrawerData(label = "Profile", icon = Icons.Filled.Person))
    drawerItemsList.add(NavigationDrawerData(label = "Favorite", icon = Icons.Filled.Favorite))
    return drawerItemsList
}
```



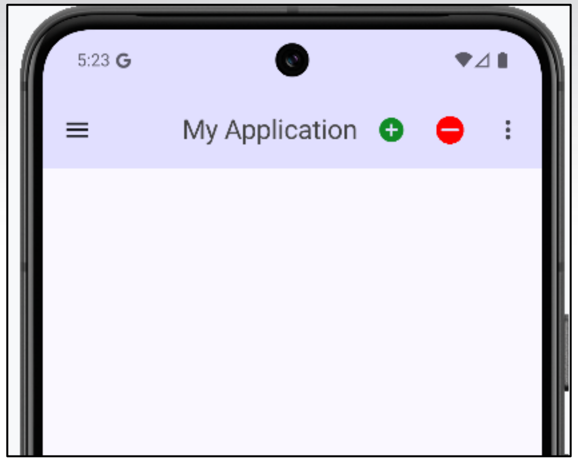


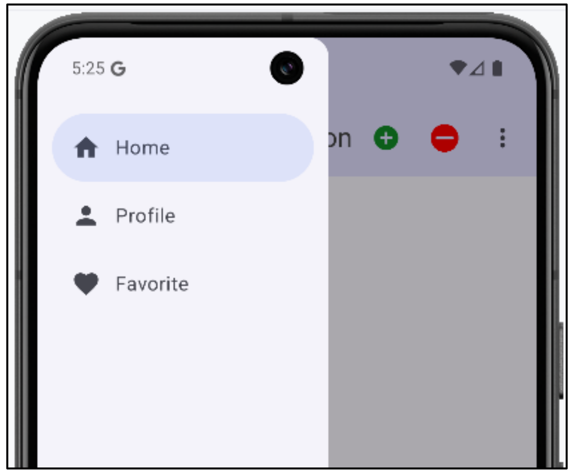
Top App Bar Navigation

Example of Top App Bar Navigation:



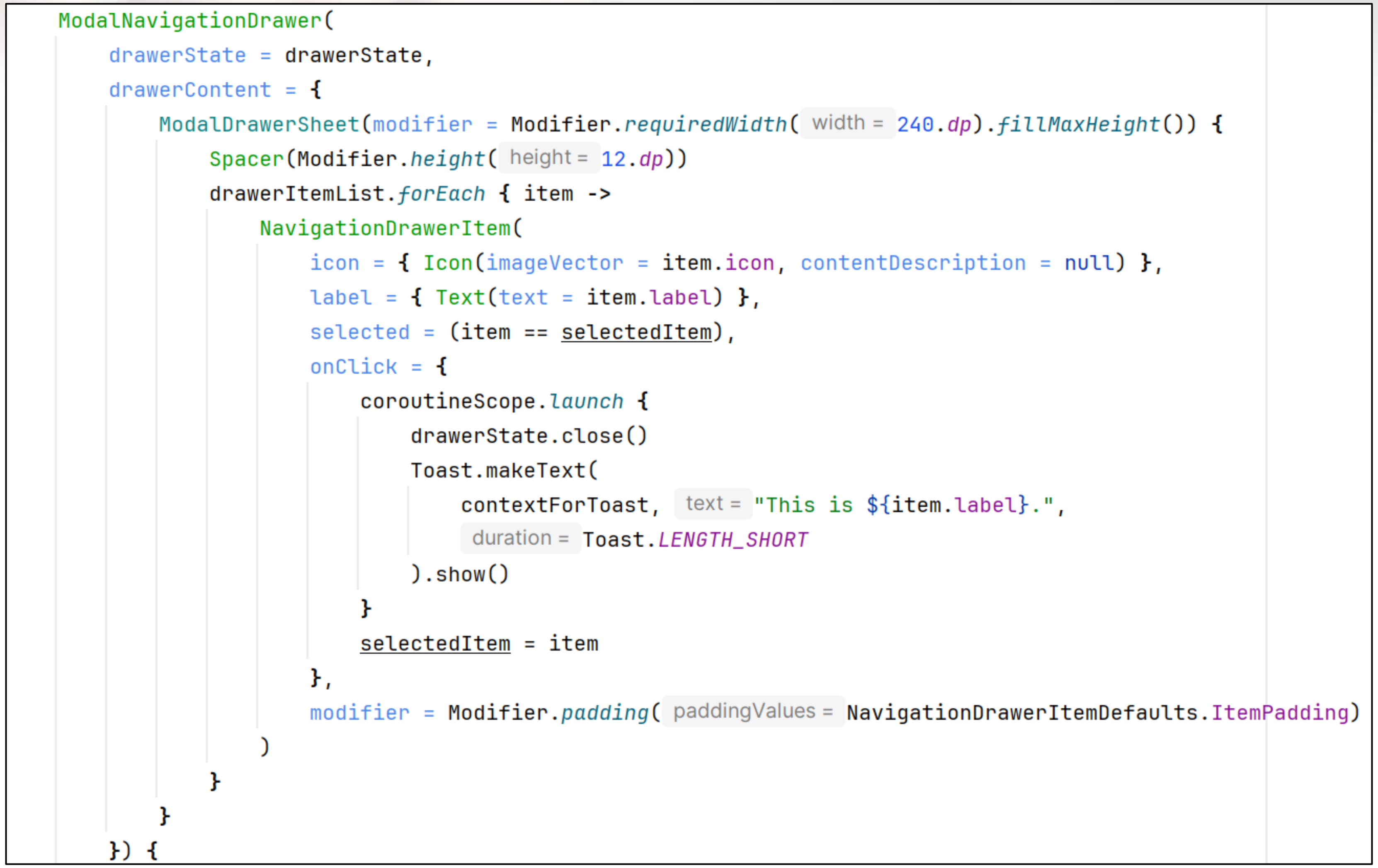
```kotlin
@OptIn( ...markerClass = ExperimentalMaterial3Api::class)
@Composable
fun MyTopBar(modifier : Modifier) {
    var expanded by remember { mutableStateOf( value = false) }
    val contextForToast = LocalContext.current.applicationContext
    val drawerState = rememberDrawerState(initialValue = DrawerValue.Closed)
    val coroutineScope = rememberCoroutineScope()
    val drawerItemList = prepareNavigationDrawerItems()
    var selectedItem by remember { mutableStateOf( value = drawerItemList[0]) }
```



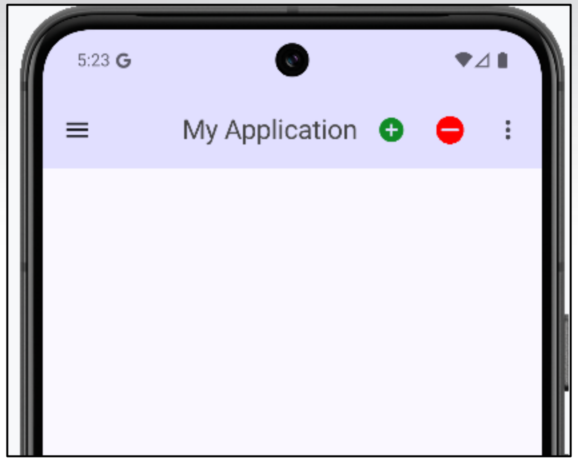


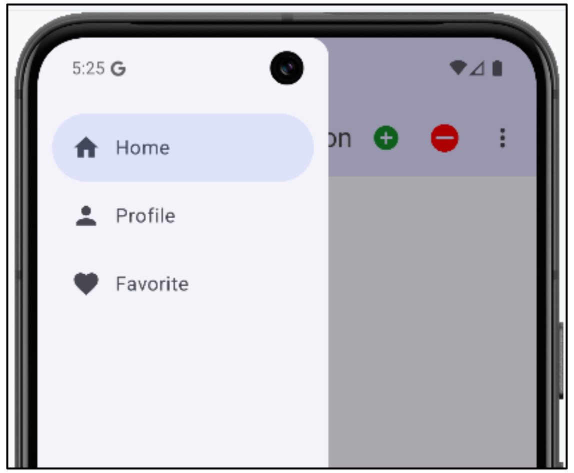
Top App Bar Navigation

Example of Top App Bar Navigation:



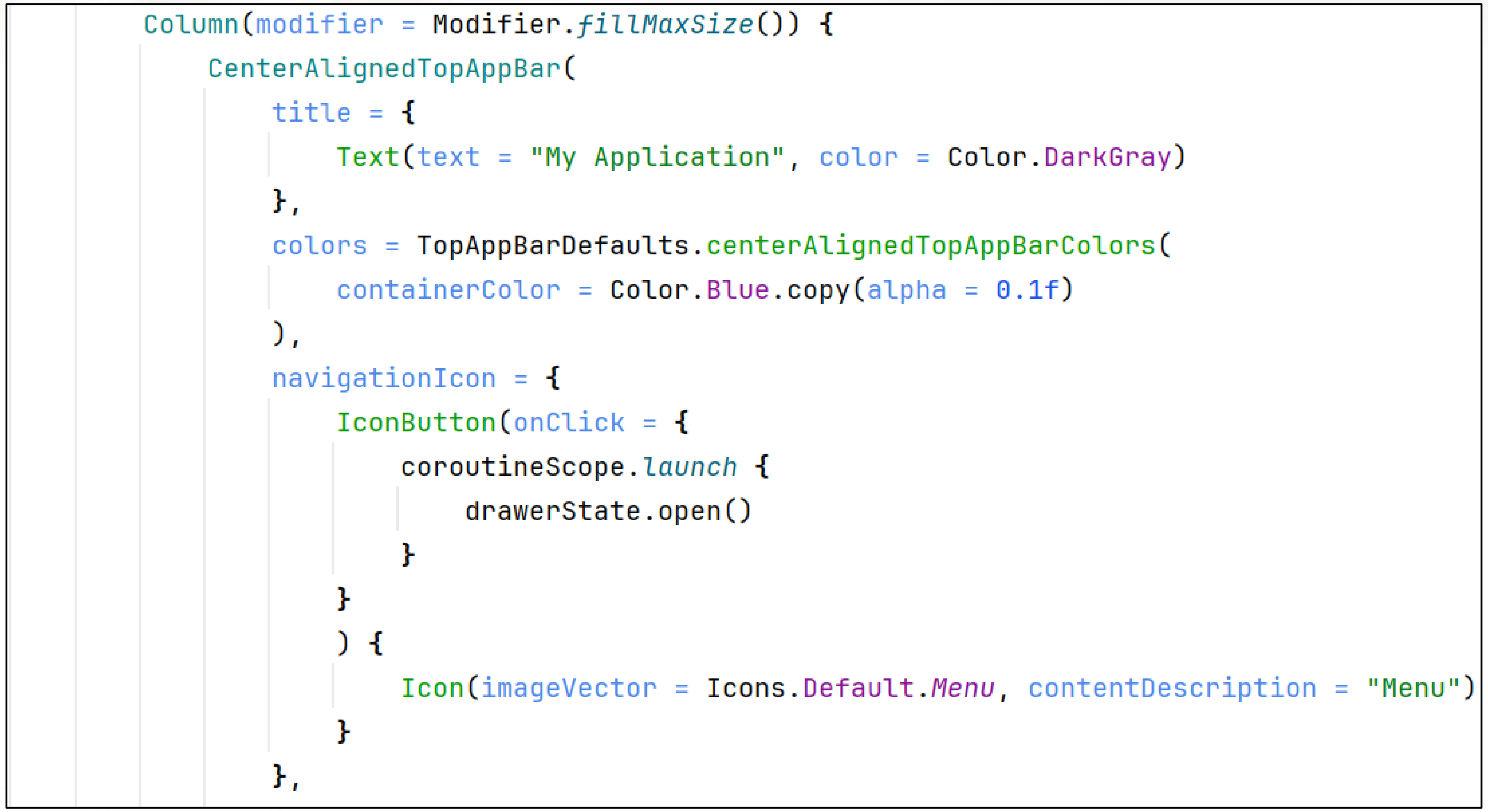
```
ModalNavigationDrawer(
    drawerState = drawerState,
    drawerContent = {
        ModalDrawerSheet(modifier = Modifier.requiredWidth( width = 240.dp).fillMaxHeight()) {
            Spacer(Modifier.height( height = 12.dp))
            drawerItemList.forEach { item ->
                NavigationDrawerItem(
                    icon = { Icon(imageVector = item.icon, contentDescription = null) },
                    label = { Text(text = item.label) },
                    selected = (item == selectedItem),
                    onClick = {
                        coroutineScope.launch {
                            drawerState.close()
                            Toast.makeText(
                                contextForToast,  text = "This is ${item.label}.",
                                 duration = Toast.LENGTH_SHORT
                            ).show()
                        }
                        selectedItem = item
                    },
                    modifier = Modifier.padding( paddingValues = NavigationDrawerItemDefaults.ItemPadding)
                )
            }
        }
    }) {
```



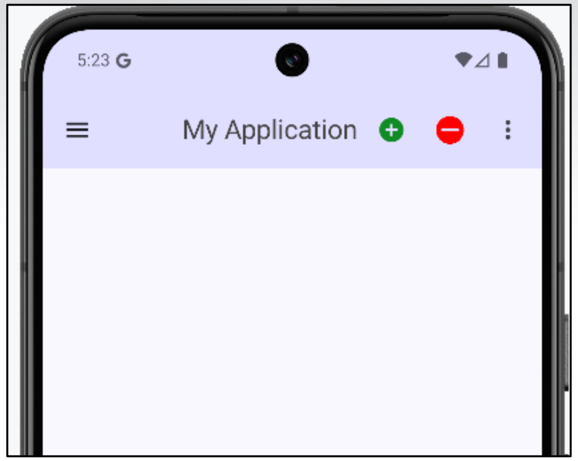


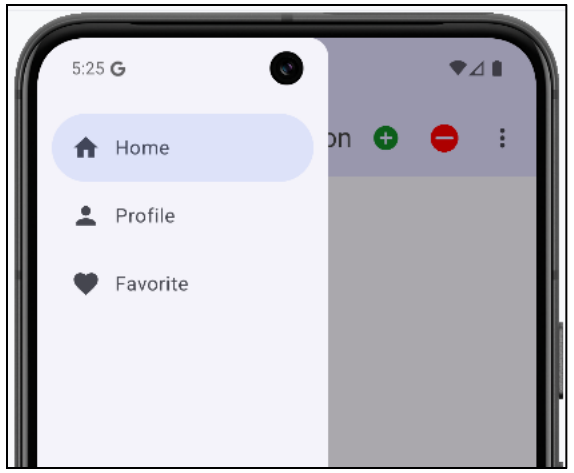
Top App Bar Navigation

Example of Top App Bar Navigation (cont.):



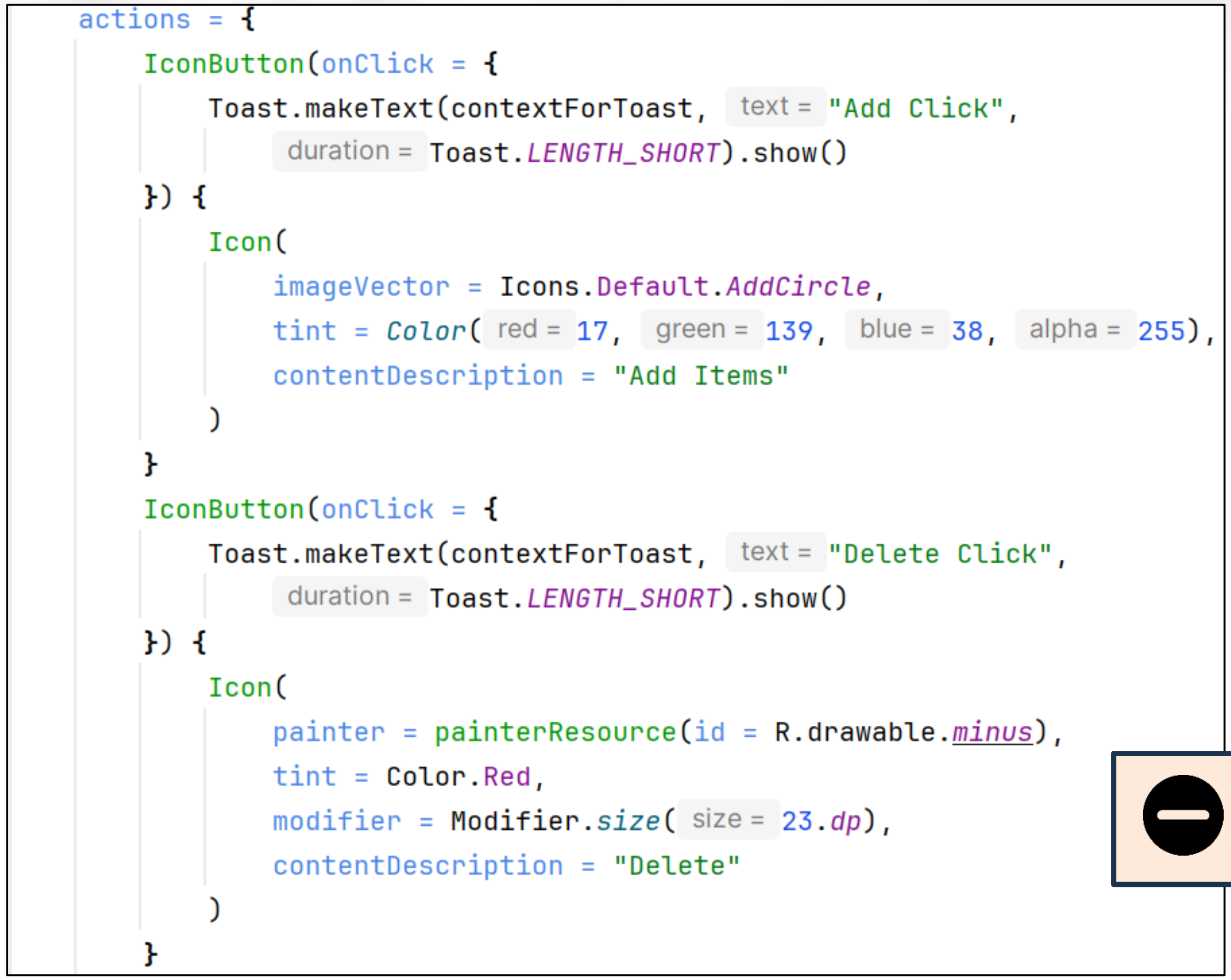
```
Column(modifier = Modifier.fillMaxSize()) {
    CenterAlignedTopAppBar(
        title = {
            Text(text = "My Application", color = Color.DarkGray)
        },
        colors = TopAppBarDefaults.centerAlignedTopAppBarColors(
            containerColor = Color.Blue.copy(alpha = 0.1f)
        ),
        navigationIcon = {
            IconButton(onClick = {
                coroutineScope.launch {
                    drawerState.open()
                }
            }
            ) {
                Icon(imageVector = Icons.Default.Menu, contentDescription = "Menu")
            }
        },
```



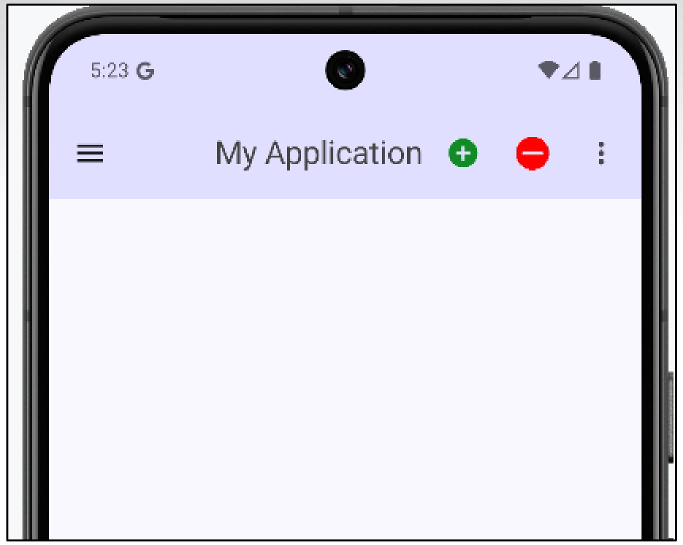


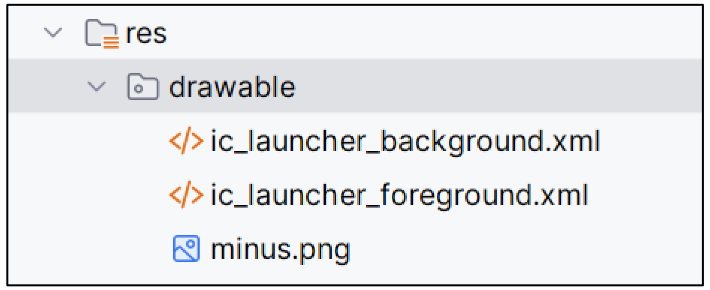
Top App Bar Navigation

Example of Top App Bar Navigation (cont.):



```
actions = {
    IconButton(onClick = {
        Toast.makeText(contextForToast,  text = "Add Click",
             duration = Toast.LENGTH_SHORT).show()
    }) {
        Icon(
            imageVector = Icons.Default.AddCircle,
            tint = Color( red = 17,  green = 139,  blue = 38,  alpha = 255),
            contentDescription = "Add Items"
        )
    }
    IconButton(onClick = {
        Toast.makeText(contextForToast,  text = "Delete Click",
             duration = Toast.LENGTH_SHORT).show()
    }) {
        Icon(
            painter = painterResource(id = R.drawable.minus),
            tint = Color.Red,
            modifier = Modifier.size( size = 23.dp),
            contentDescription = "Delete"
        )
    }
```

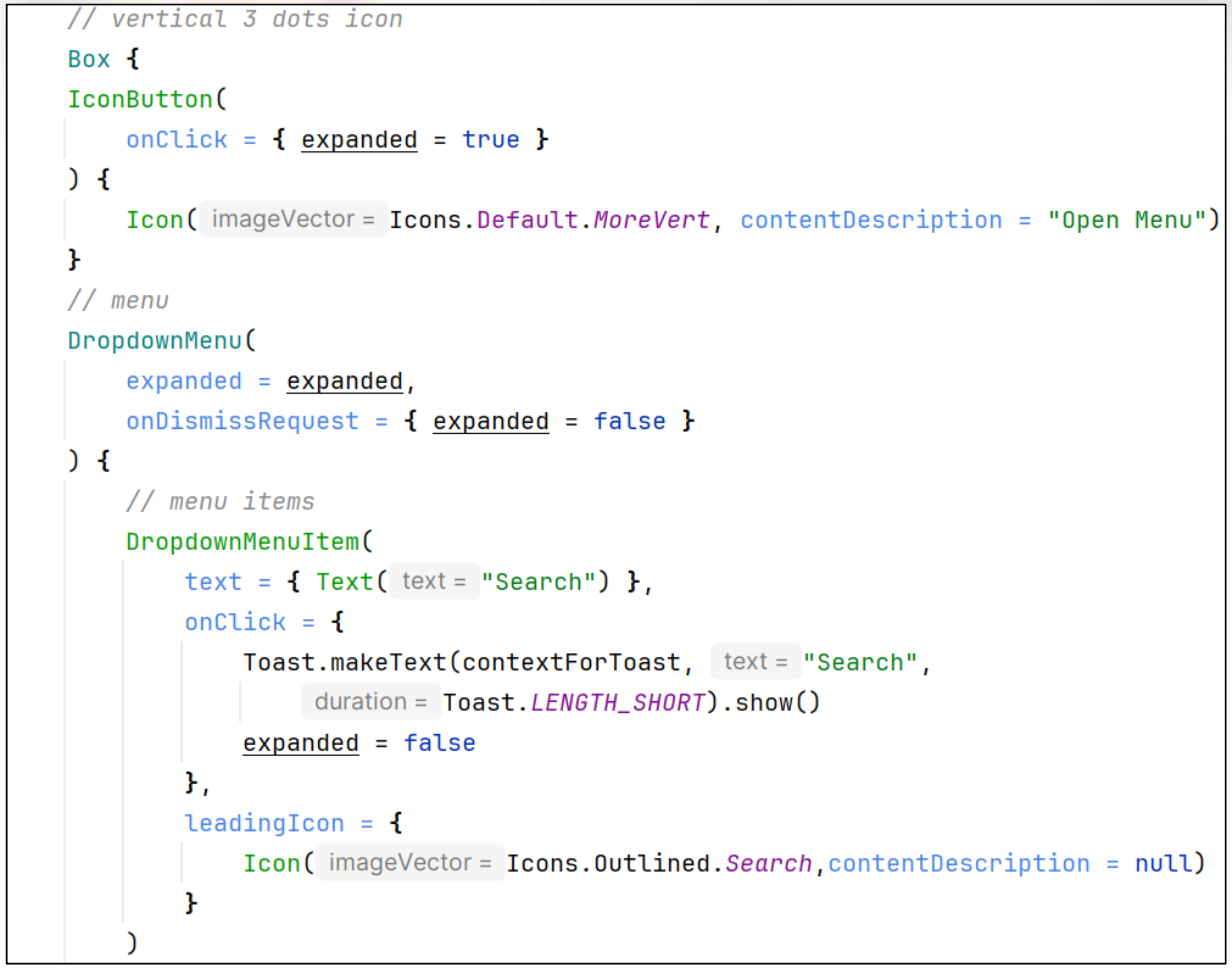




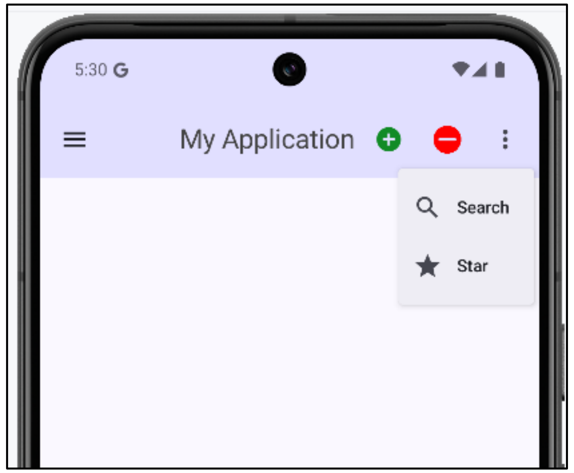
```
res
  drawable
    ic_launcher_background.xml
    ic_launcher_foreground.xml
    minus.png
```

Top App Bar Navigation

Example of Top App Bar Navigation (cont.):

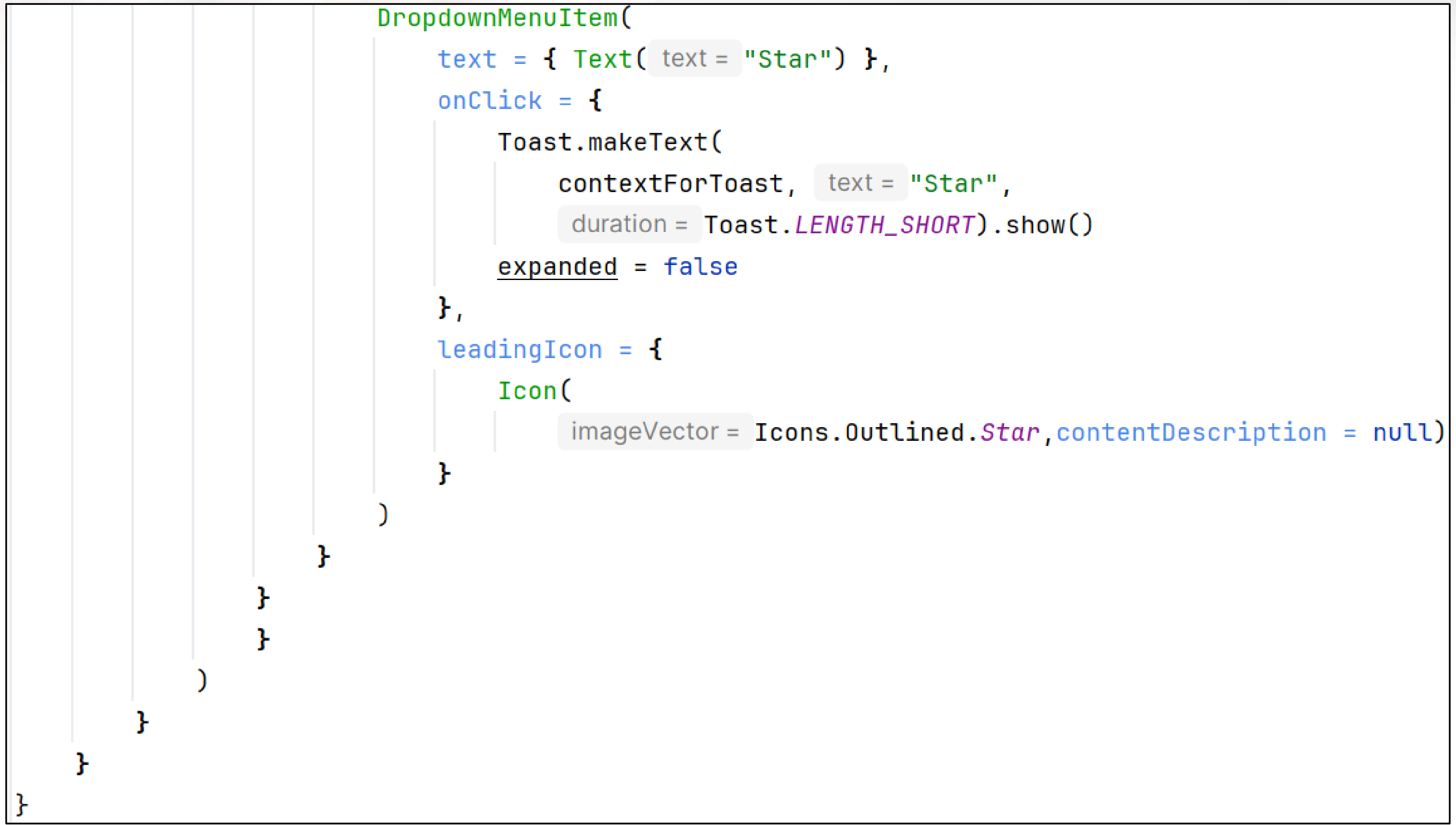


```
// vertical 3 dots icon
Box {
IconButton(
    onClick = { expanded = true }
) {
    Icon( imageVector = Icons.Default.MoreVert, contentDescription = "Open Menu")
}
// menu
DropdownMenu(
    expanded = expanded,
    onDismissRequest = { expanded = false }
) {
    // menu items
    DropdownMenuItem(
        text = { Text( text = "Search") },
        onClick = {
            Toast.makeText(contextForToast,  text = "Search",
                 duration = Toast.LENGTH_SHORT).show()
            expanded = false
        },
        leadingIcon = {
            Icon( imageVector = Icons.Outlined.Search,contentDescription = null)
        }
    )
```

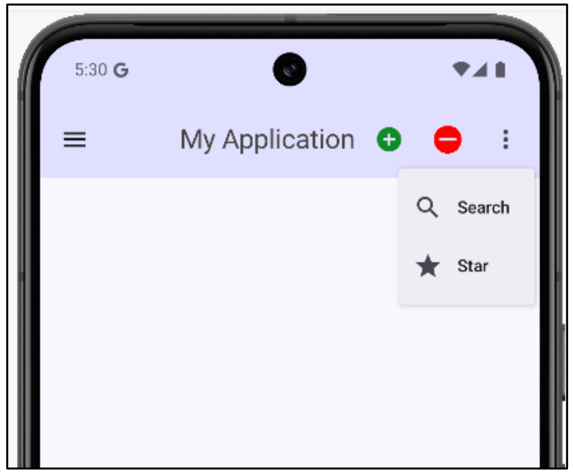


Top App Bar Navigation

Example of Top App Bar Navigation (cont.):



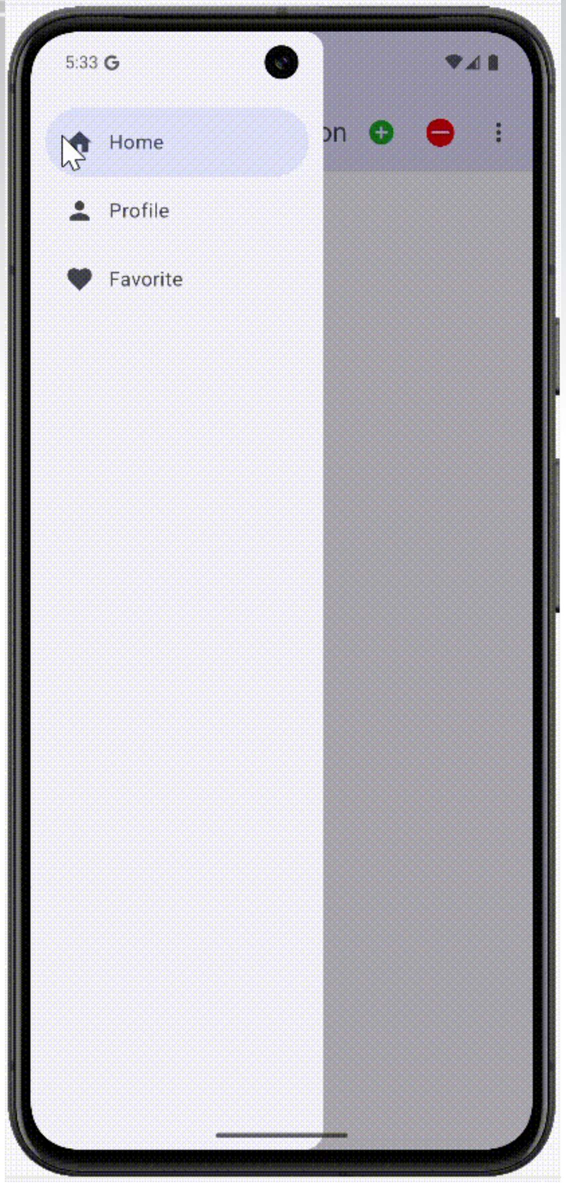
```
                        DropdownMenuItem(
                            text = { Text( text = "Star") },
                            onClick = {
                                Toast.makeText(
                                    contextForToast,  text = "Star",
                                     duration = Toast.LENGTH_SHORT).show()
                                expanded = false
                            },
                            leadingIcon = {
                                Icon(
                                     imageVector = Icons.Outlined.Star,contentDescription = null)
                            }
                        )
                    }
                }
                }
            )
        }
    }
}
```



Top App Bar Navigation

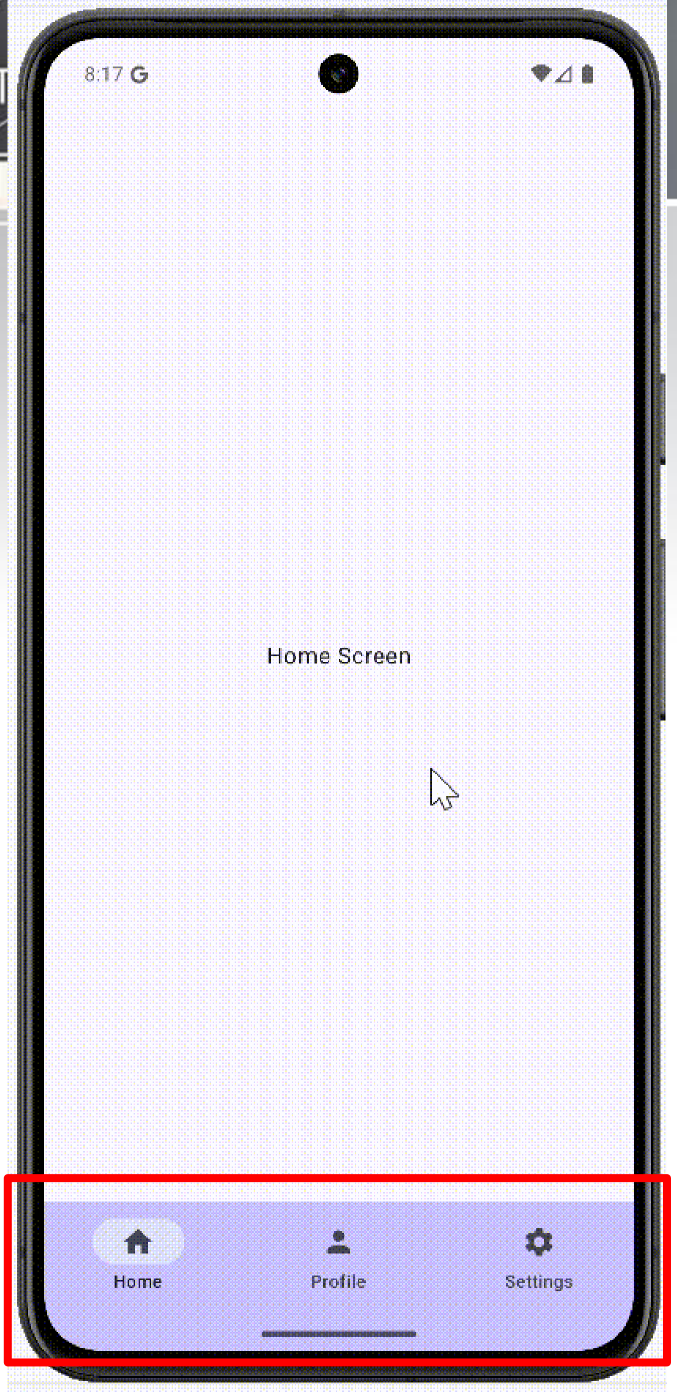
Example of Top App Bar Navigation (cont.):

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            Lec6DrawerTheme {
                Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->
                    MyTopBar(modifier = Modifier.padding( paddingValues = innerPadding))
                }
            }
        }
    }
}
```



Bottom Navigation Bar

- BottomNavigationBar is a widget that displays a row of small widgets at the bottom of an app.
- Usually, show around three to five items.
- Each item must have a label and an icon.
- BottomNavigationBar allows to select one item at a time and quickly navigate to a given page.



Bottom Navigation Bar

Component of files in project

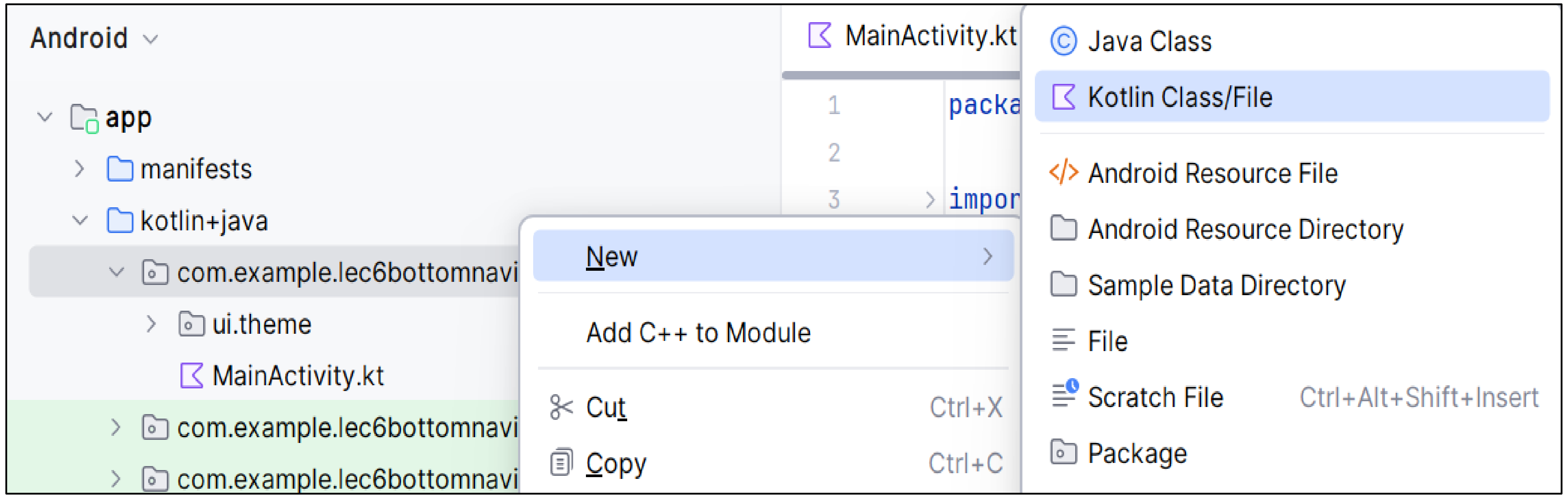
- HomeScreen.kt
- ProfileScreen.kt
- SettingsScreen.kt

Composable functions that define the **UI layout** for each screen.

- Screen.kt (sealed class) Defines the routes (URLs/Paths) for each screen.
- NavGraph.kt The central hub that **connects Routes to Screens**.

Bottom Navigation Bar

- Create the following HomeScreen.kt, ProfileScreen.kt and SettingsScreen.kt



Android

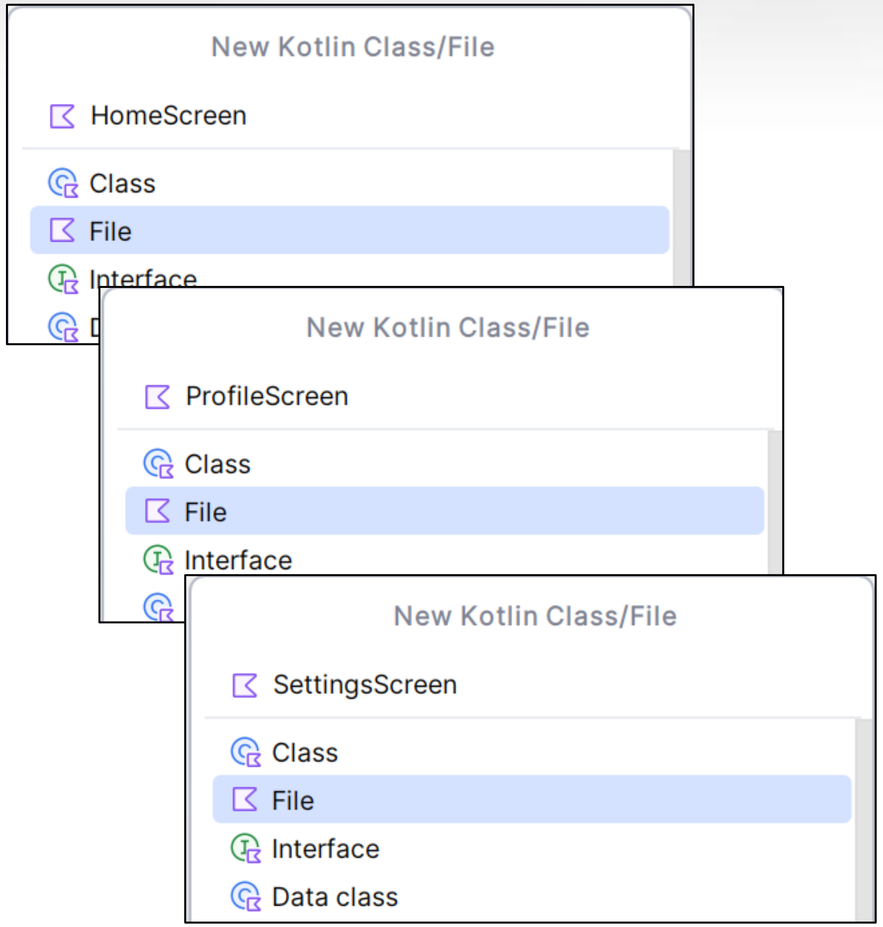
app
manifests
kotlin+java
com.example.lec6bottomnavi
ui.theme
MainActivity.kt
com.example.lec6bottomnavi
com.example.lec6bottomnavi

MainActivity.kt

```
packa
impor
```

New
Add C++ to Module
Cut Ctrl+X
Copy Ctrl+C

Java Class
Kotlin Class/File
Android Resource File
Android Resource Directory
Sample Data Directory
File
Scratch File Ctrl+Alt+Shift+Insert
Package



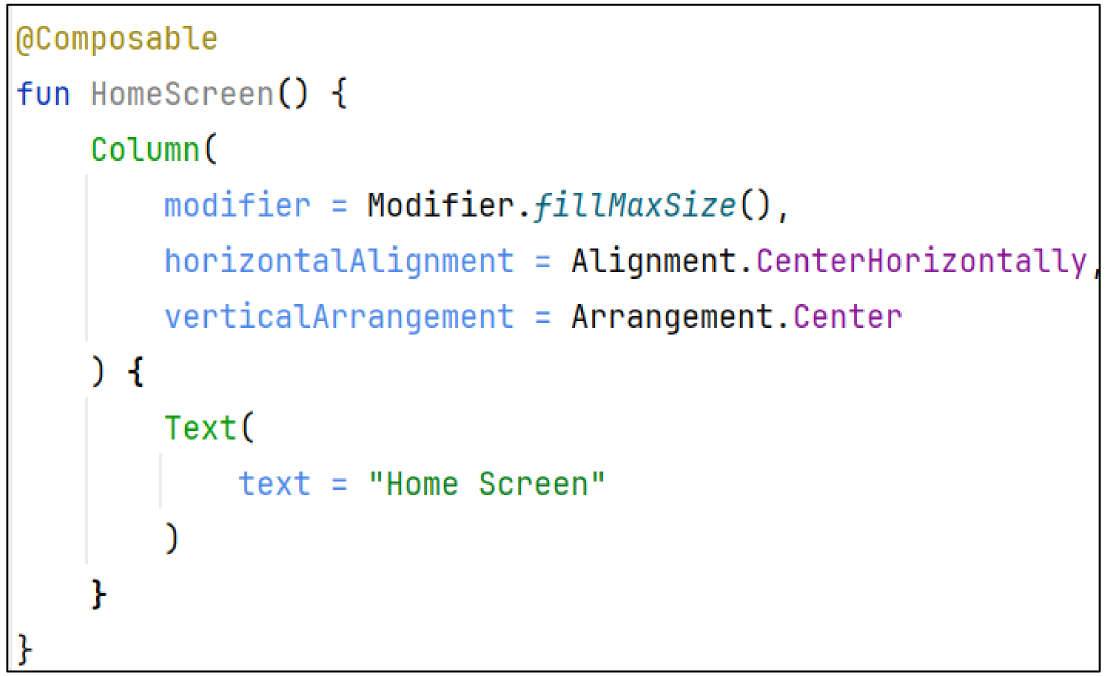
New Kotlin Class/File
HomeScreen
Class
File
Interface

New Kotlin Class/File
ProfileScreen
Class
File
Interface

New Kotlin Class/File
SettingsScreen
Class
File
Interface
Data class

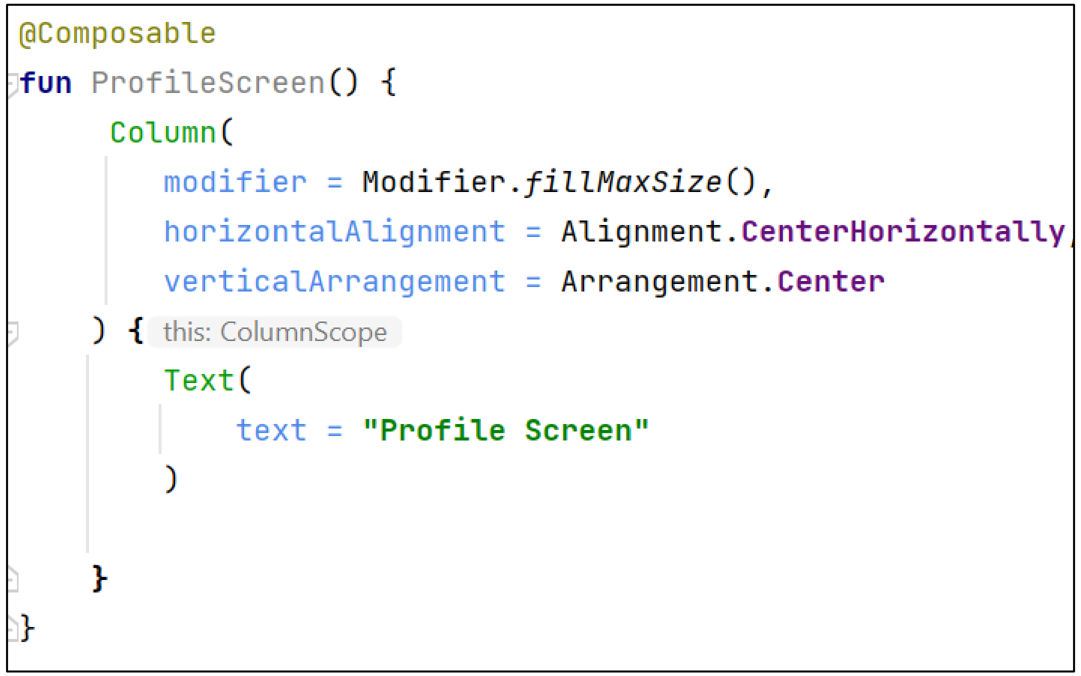
Bottom Navigation Bar

- Add command to HomeScreen.kt, ProfileScreen.kt and SettingsScreen.kt file



```kotlin
@Composable
fun HomeScreen() {
    Column(
        modifier = Modifier.fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {
        Text(
            text = "Home Screen"
        )
    }
}
```

HomeScreen.kt file

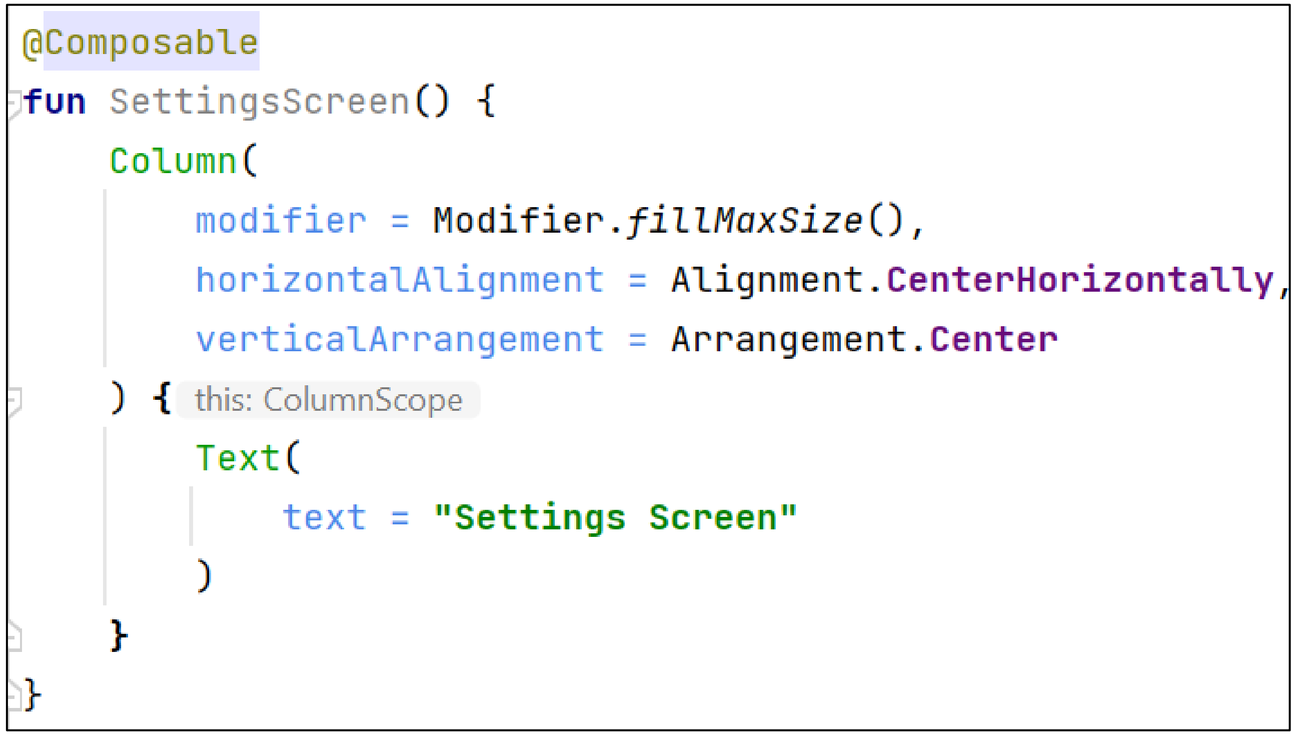


```kotlin
@Composable
fun ProfileScreen() {
    Column(
        modifier = Modifier.fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) { this: ColumnScope
        Text(
            text = "Profile Screen"
        )

    }
}
```

ProfileScreen.kt file

Bottom Navigation Bar

- Add command to HomeScreen.kt, ProfileScreen.kt and SettingsScreen.kt file

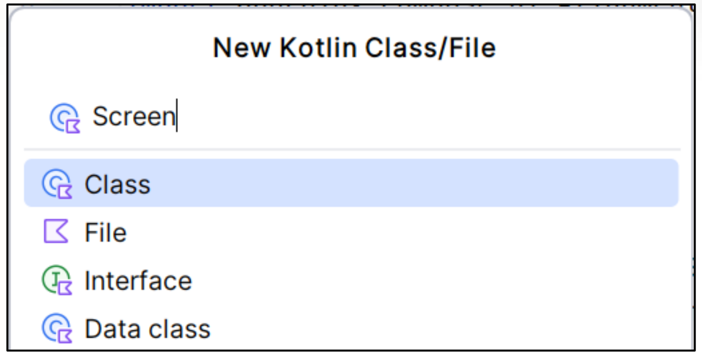


```
@Composable
fun SettingsScreen() {
    Column(
        modifier = Modifier.fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) { this: ColumnScope
        Text(
            text = "Settings Screen"
        )
    }
}
```

SettingsScreen.kt file

Bottom Navigation Bar

- Create class Screen.kt and add this code.

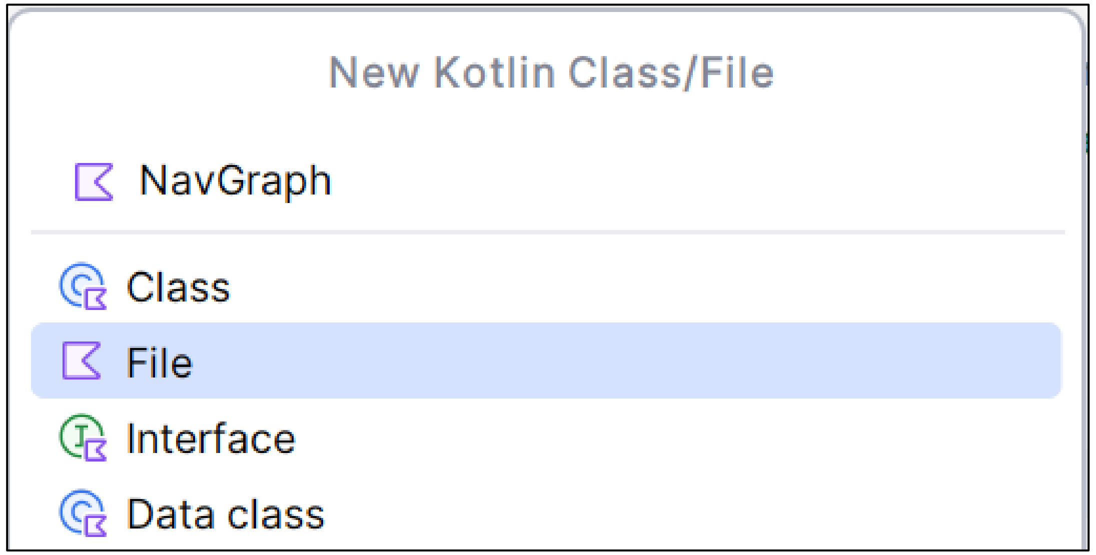


Screen.kt File

```
sealed class Screen(val route: String, val name: String, val icon: ImageVector) {
    data  object Home: Screen(route = "home_screen", name = "Home", icon = Icons.Default.Home)
    data object Profile: Screen(route = "profile_screen", name = "Profile", icon = Icons.Default.Person)
    data object Settings: Screen(route = "settings_screen", name = "Settings", icon = Icons.Default.Settings)
}
```

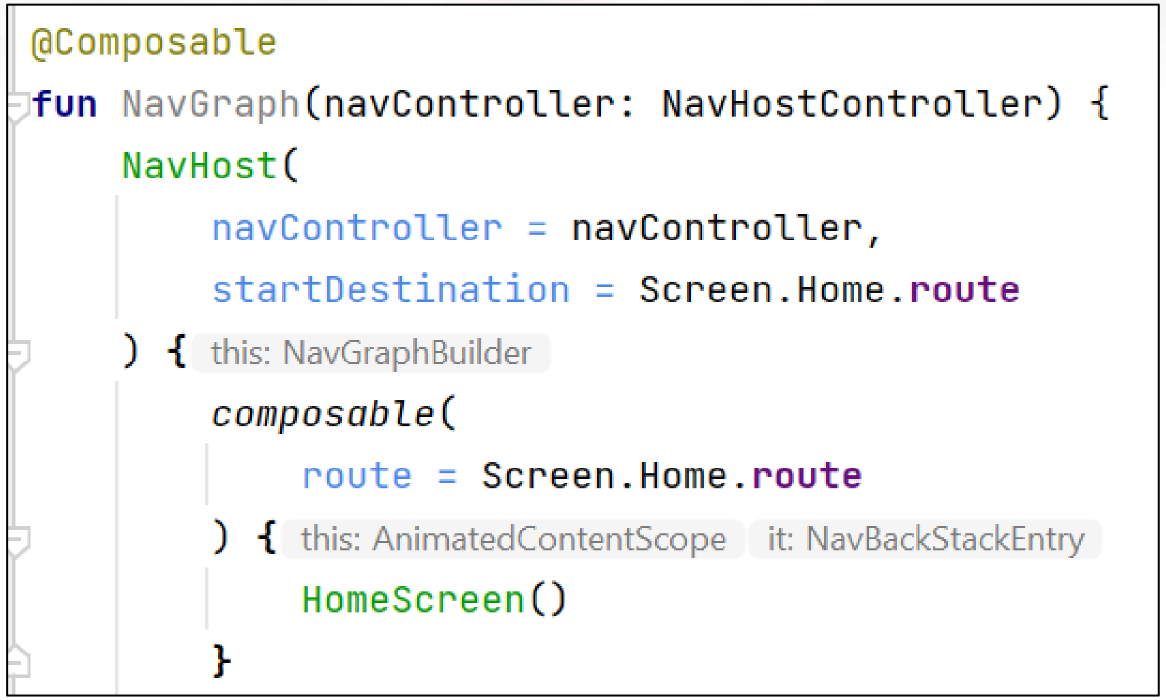
Bottom Navigation Bar

- Next, set the navigation graph. It contains all the screens (destinations) along with the logical connections between them.
- Create the NavGraph.kt file and add the following code:

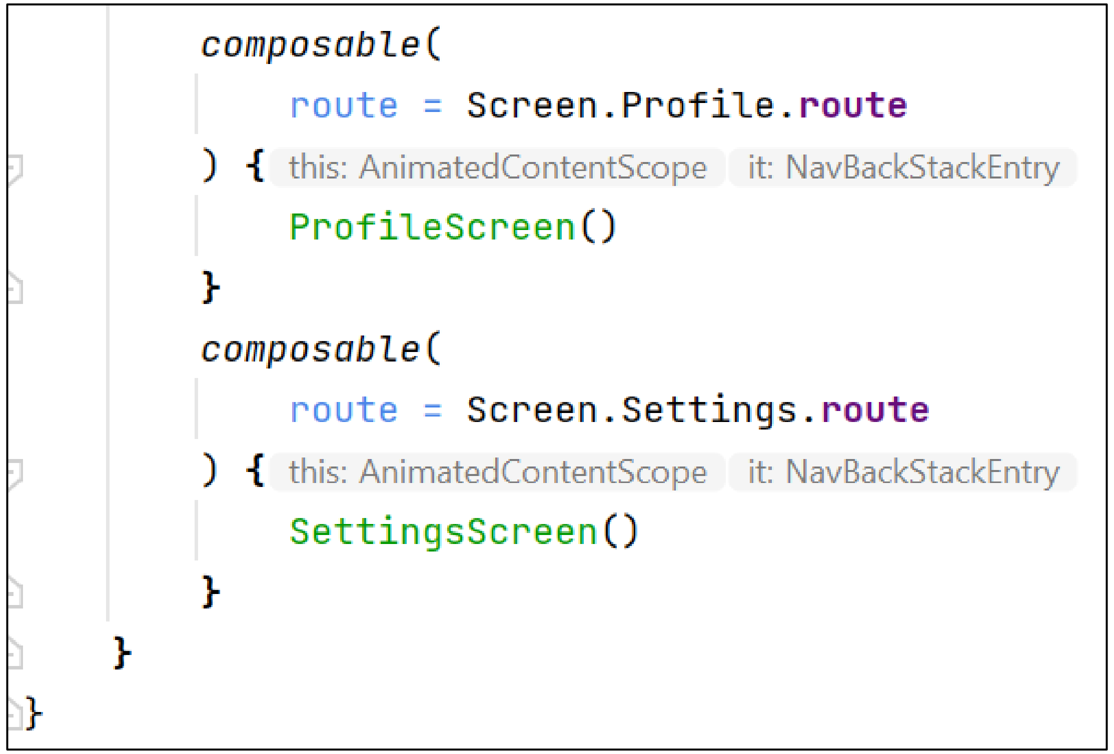


Bottom Navigation Bar

The NavGraph.kt file is added with the following code:



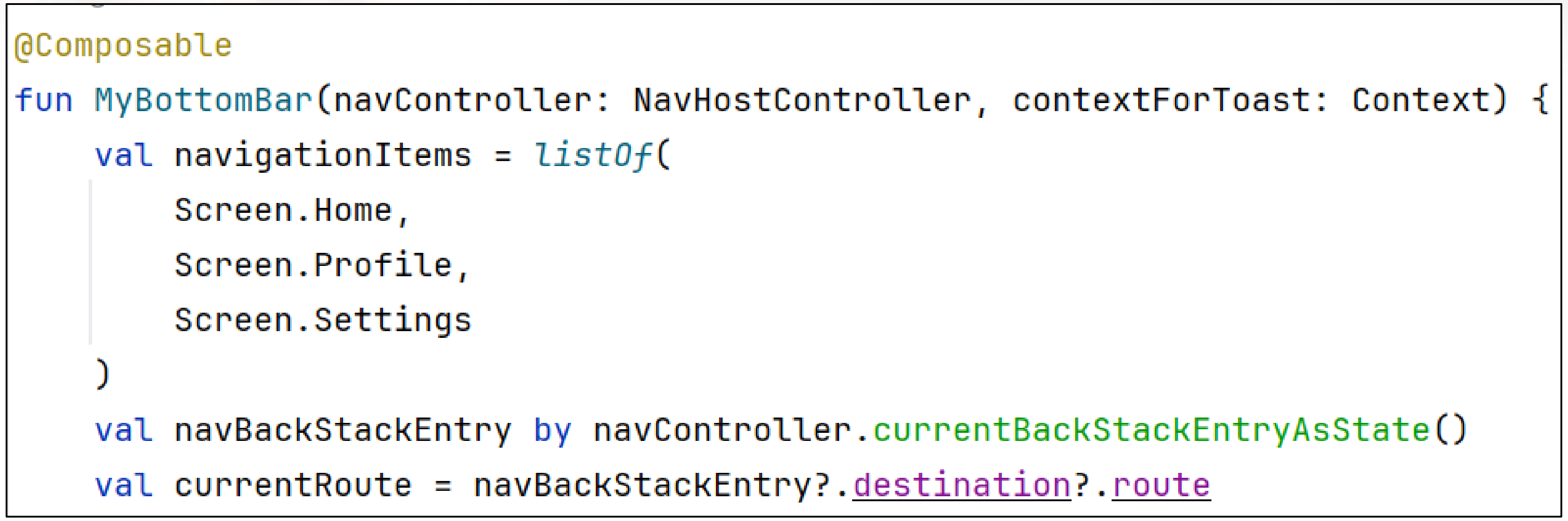
```
@Composable
fun NavGraph(navController: NavHostController) {
    NavHost(
        navController = navController,
        startDestination = Screen.Home.route
    ) { this: NavGraphBuilder
        composable(
            route = Screen.Home.route
        ) { this: AnimatedContentScope  it: NavBackStackEntry
            HomeScreen()
        }
```



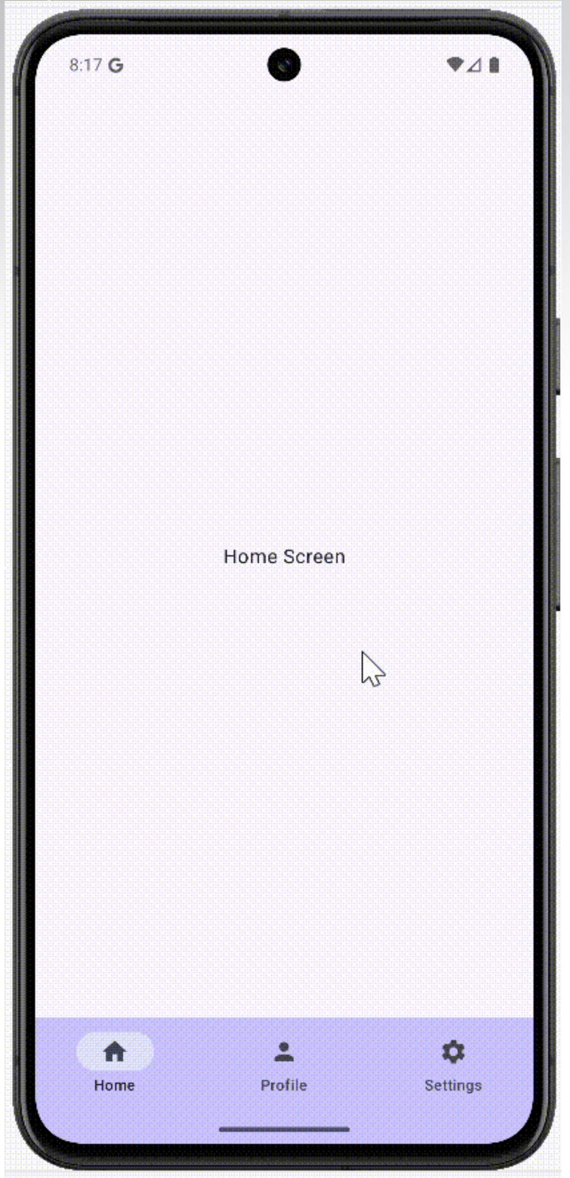
```
        composable(
            route = Screen.Profile.route
        ) { this: AnimatedContentScope  it: NavBackStackEntry
            ProfileScreen()
        }
        composable(
            route = Screen.Settings.route
        ) { this: AnimatedContentScope  it: NavBackStackEntry
            SettingsScreen()
        }
    }
}
```

Bottom Navigation Bar

- Next, go to MainActivity.kt file and add MyBottomBar function

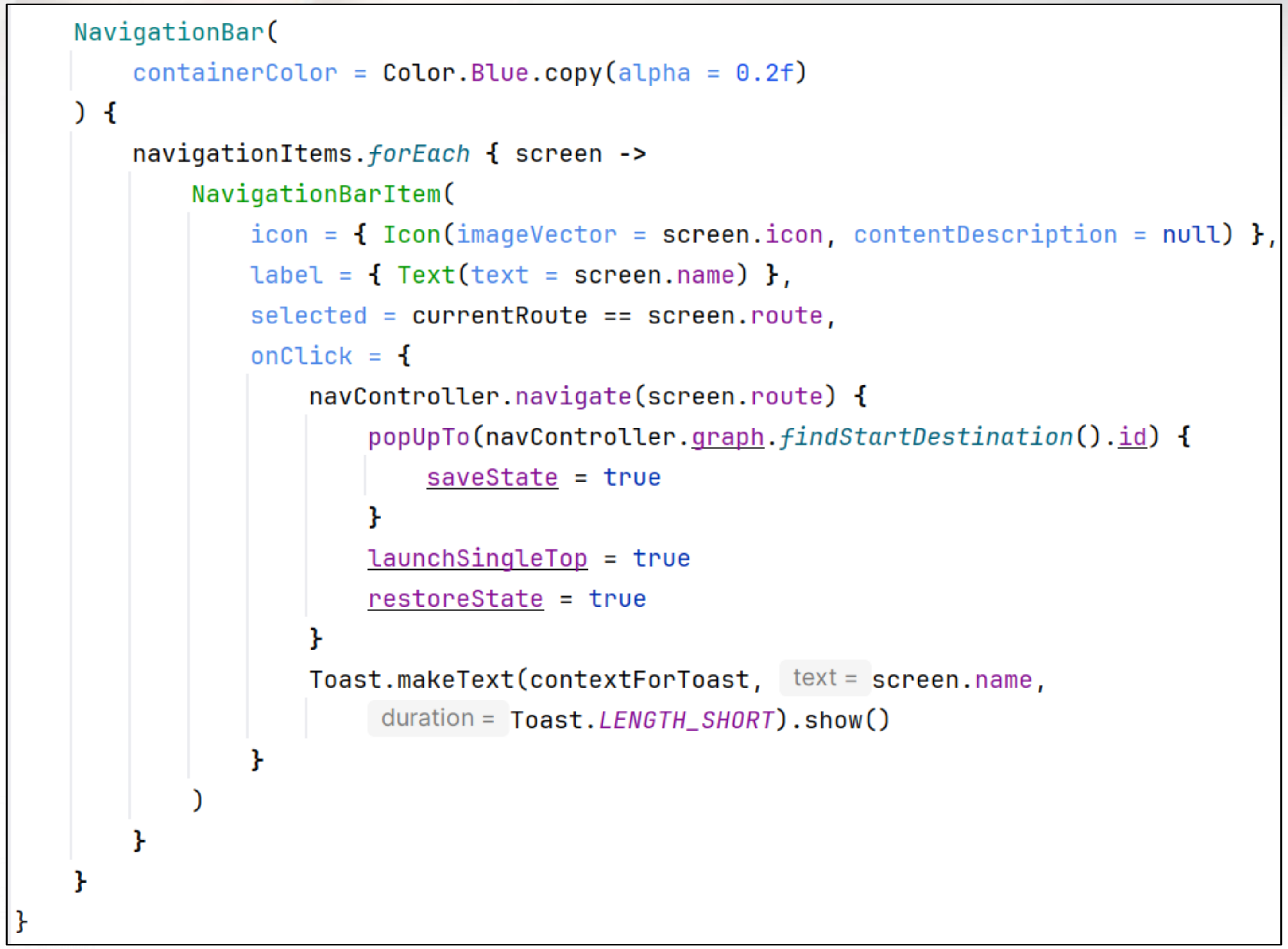


```kotlin
@Composable
fun MyBottomBar(navController: NavHostController, contextForToast: Context) {
    val navigationItems = listOf(
        Screen.Home,
        Screen.Profile,
        Screen.Settings
    )
    val navBackStackEntry by navController.currentBackStackEntryAsState()
    val currentRoute = navBackStackEntry?.destination?.route
```

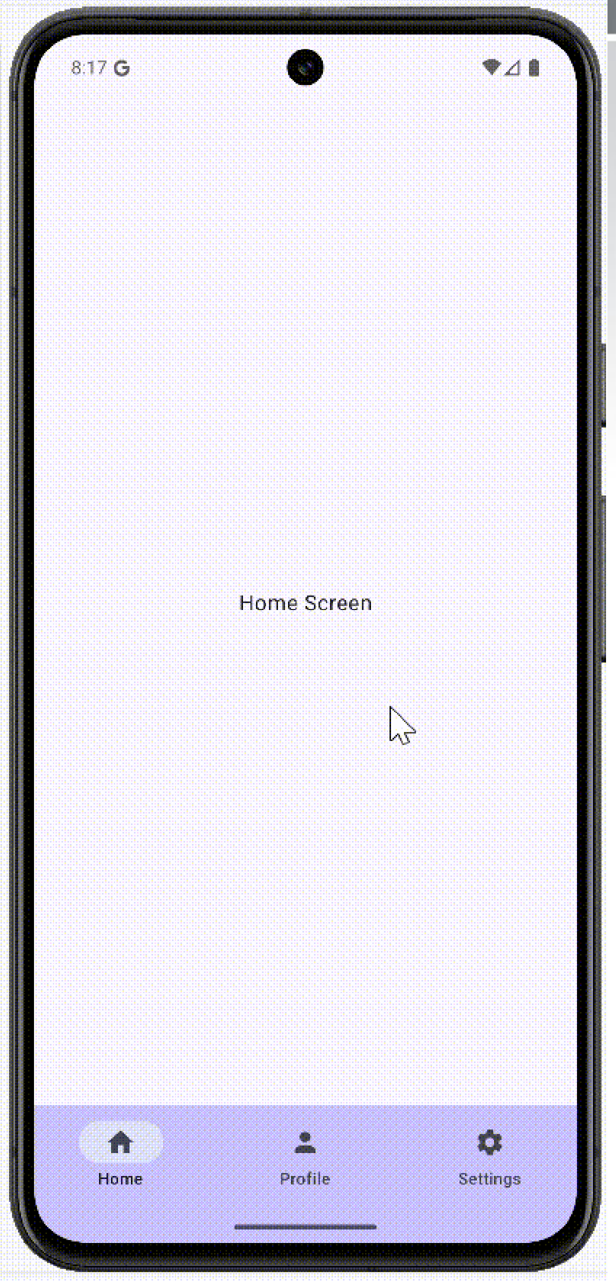


Bottom Navigation Bar

- MainActivity and add MyBottomBar function(cont.)



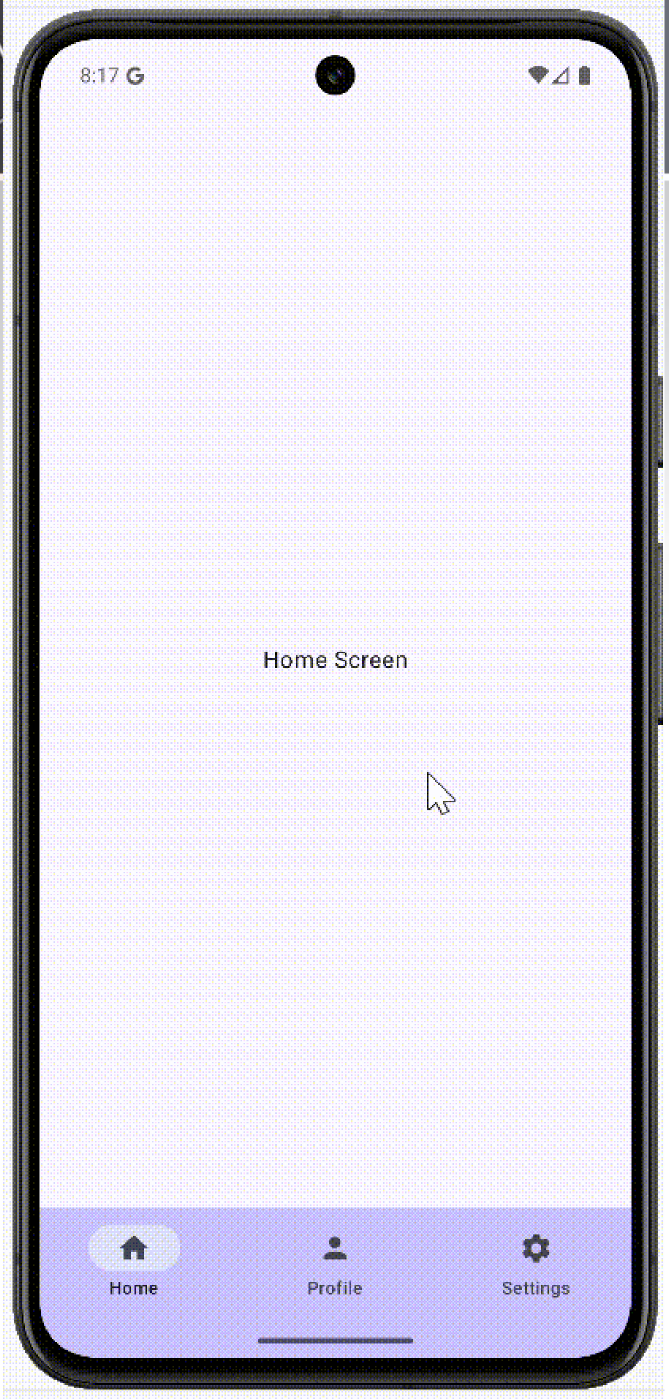
```
    NavigationBar(
        containerColor = Color.Blue.copy(alpha = 0.2f)
    ) {
        navigationItems.forEach { screen ->
            NavigationBarItem(
                icon = { Icon(imageVector = screen.icon, contentDescription = null) },
                label = { Text(text = screen.name) },
                selected = currentRoute == screen.route,
                onClick = {
                    navController.navigate(screen.route) {
                        popUpTo(navController.graph.findStartDestination().id) {
                            saveState = true
                        }
                        launchSingleTop = true
                        restoreState = true
                    }
                    Toast.makeText(contextForToast, text = screen.name,
                        duration = Toast.LENGTH_SHORT).show()
                }
            )
        }
    }
}
```



Bottom Navigation Bar

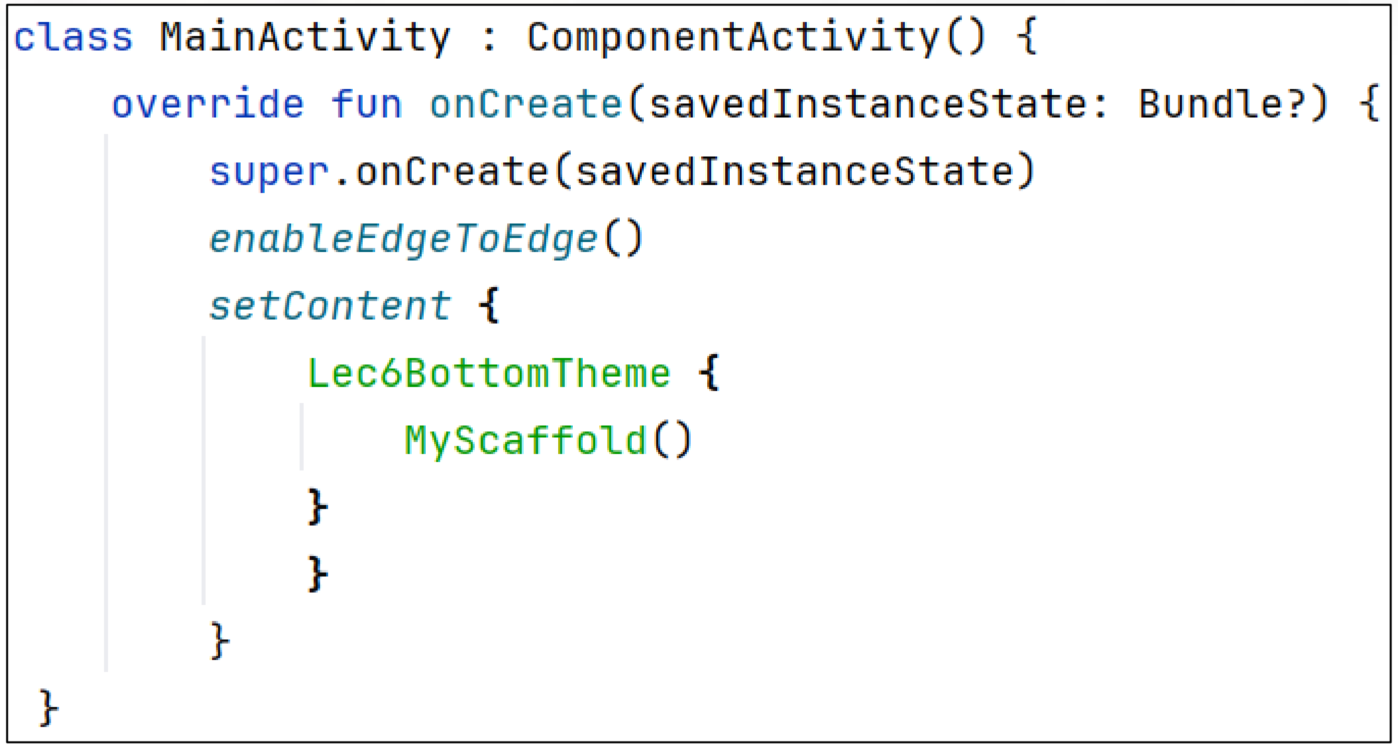
- MainActivity and add MyScaffold function (cont.)

```
@Composable
fun MyScaffold() {
    val contextForToast = LocalContext.current.applicationContext
    val navController = rememberNavController()

    Scaffold(
        topBar = { },
        bottomBar = { MyBottomBar(navController, contextForToast) },
    ) { paddingValues ->
        Column(
            modifier = Modifier
                .fillMaxSize()
                .padding(paddingValues)
        ) {
            // Call NavGraph
            NavGraph(navController = navController)
        }
    }
}
```

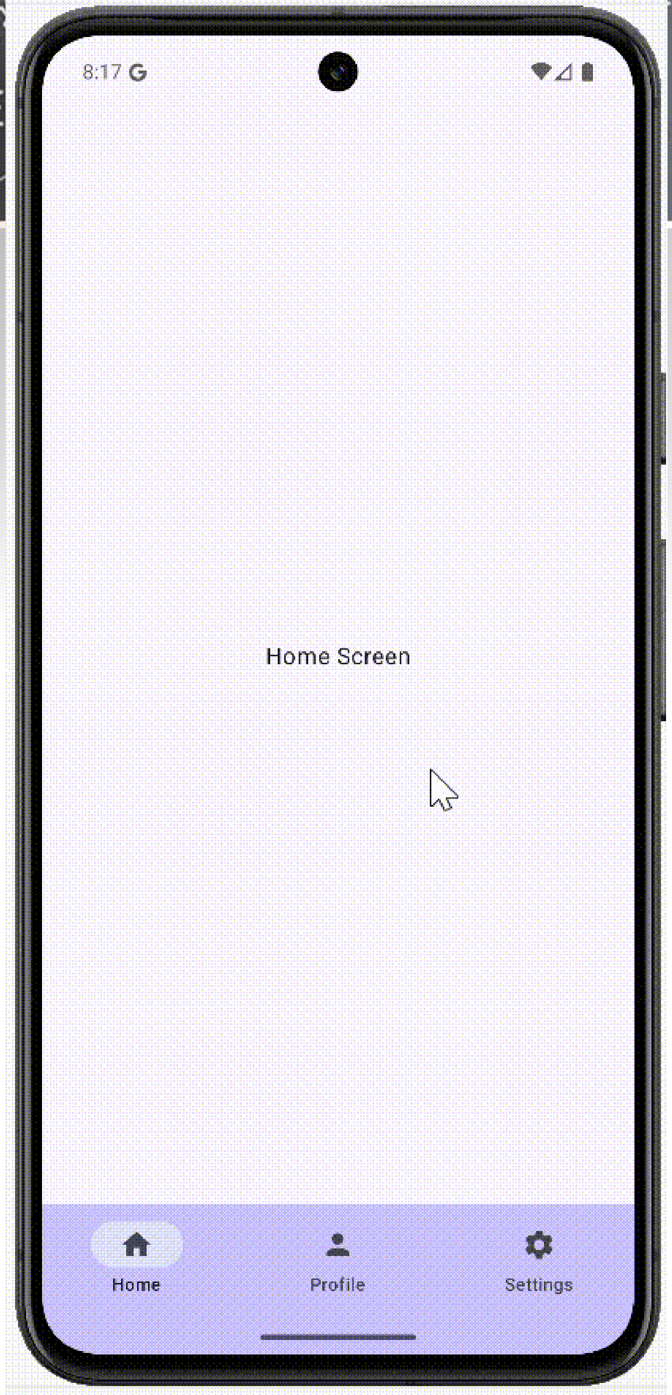


Bottom Navigation Bar

- MainActivity and add the following code (cont.)

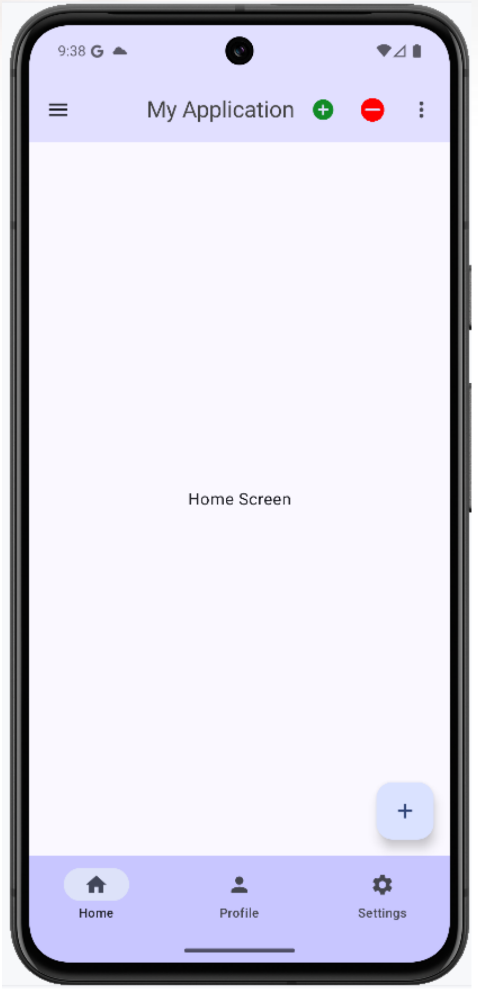


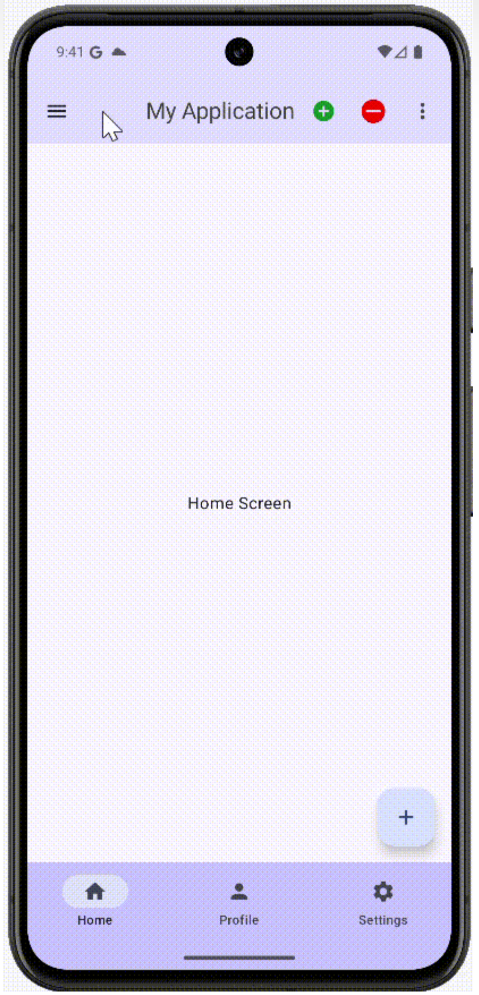
```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            Lec6BottomTheme {
                MyScaffold()
            }
            }
        }
 }
```



Scaffold & Navigation Components

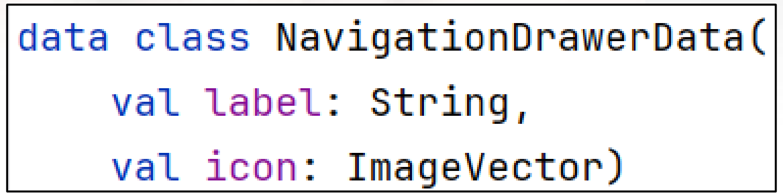
Top Bar + Drawer + Bottom Navigation + Scaffold





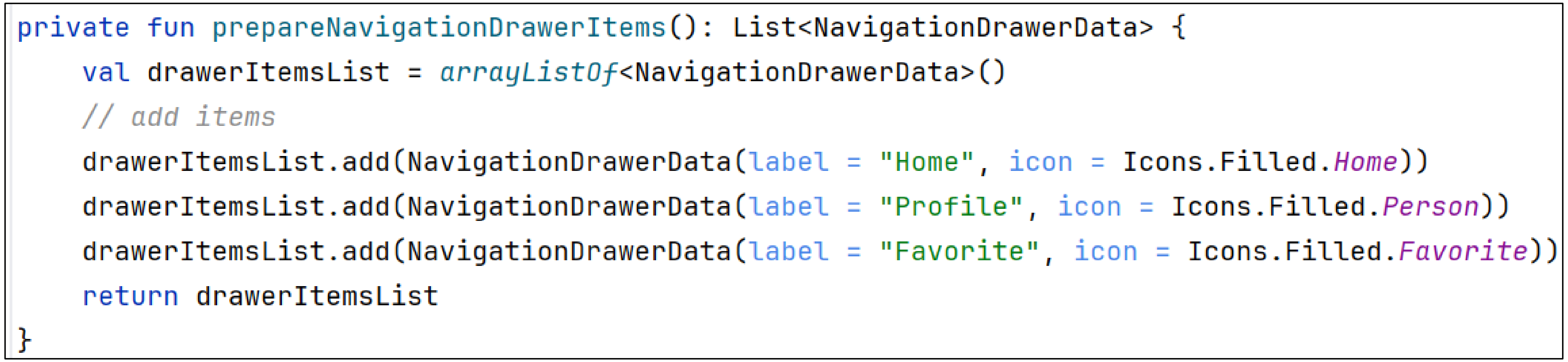
Scaffold & Navigation Components

data class: NavigationDrawerData



```kotlin
data class NavigationDrawerData(
    val label: String,
    val icon: ImageVector)
```

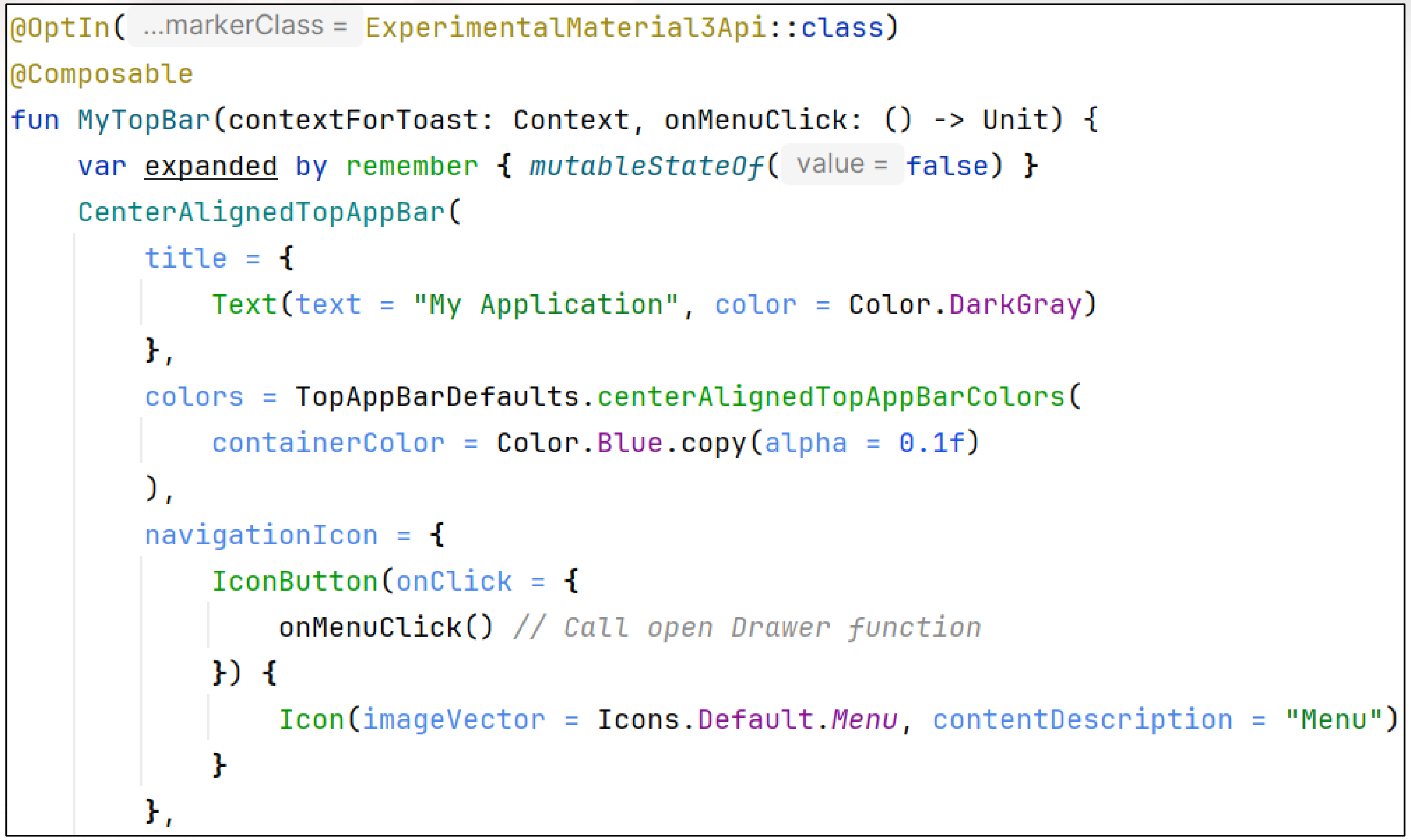
Function: prepareNavigationDrawerItems()



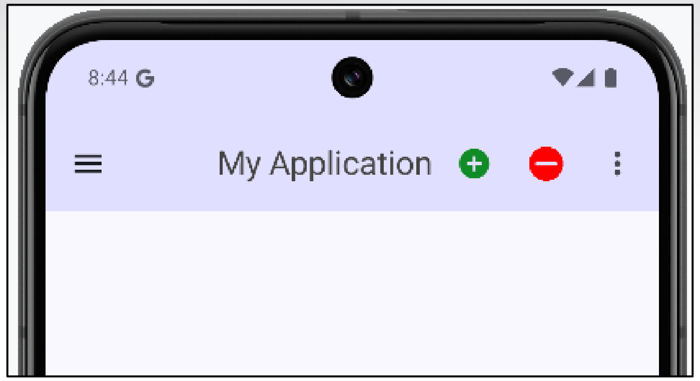
```kotlin
private fun prepareNavigationDrawerItems(): List<NavigationDrawerData> {
    val drawerItemsList = arrayListOf<NavigationDrawerData>()
    // add items
    drawerItemsList.add(NavigationDrawerData(label = "Home", icon = Icons.Filled.Home))
    drawerItemsList.add(NavigationDrawerData(label = "Profile", icon = Icons.Filled.Person))
    drawerItemsList.add(NavigationDrawerData(label = "Favorite", icon = Icons.Filled.Favorite))
    return drawerItemsList
}
```

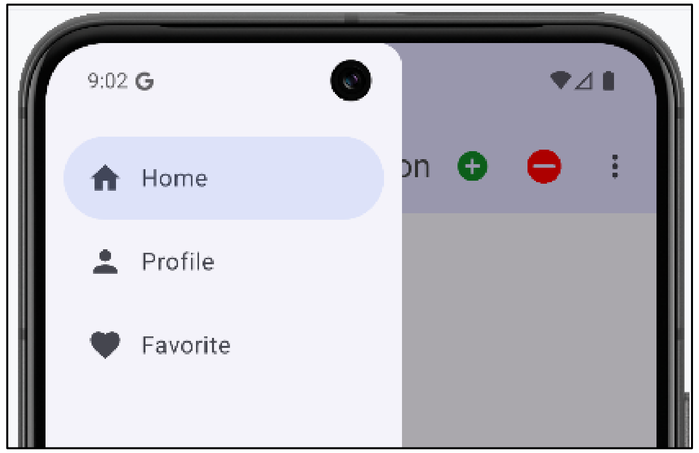
Scaffold & Navigation Components

MyTopBar Function



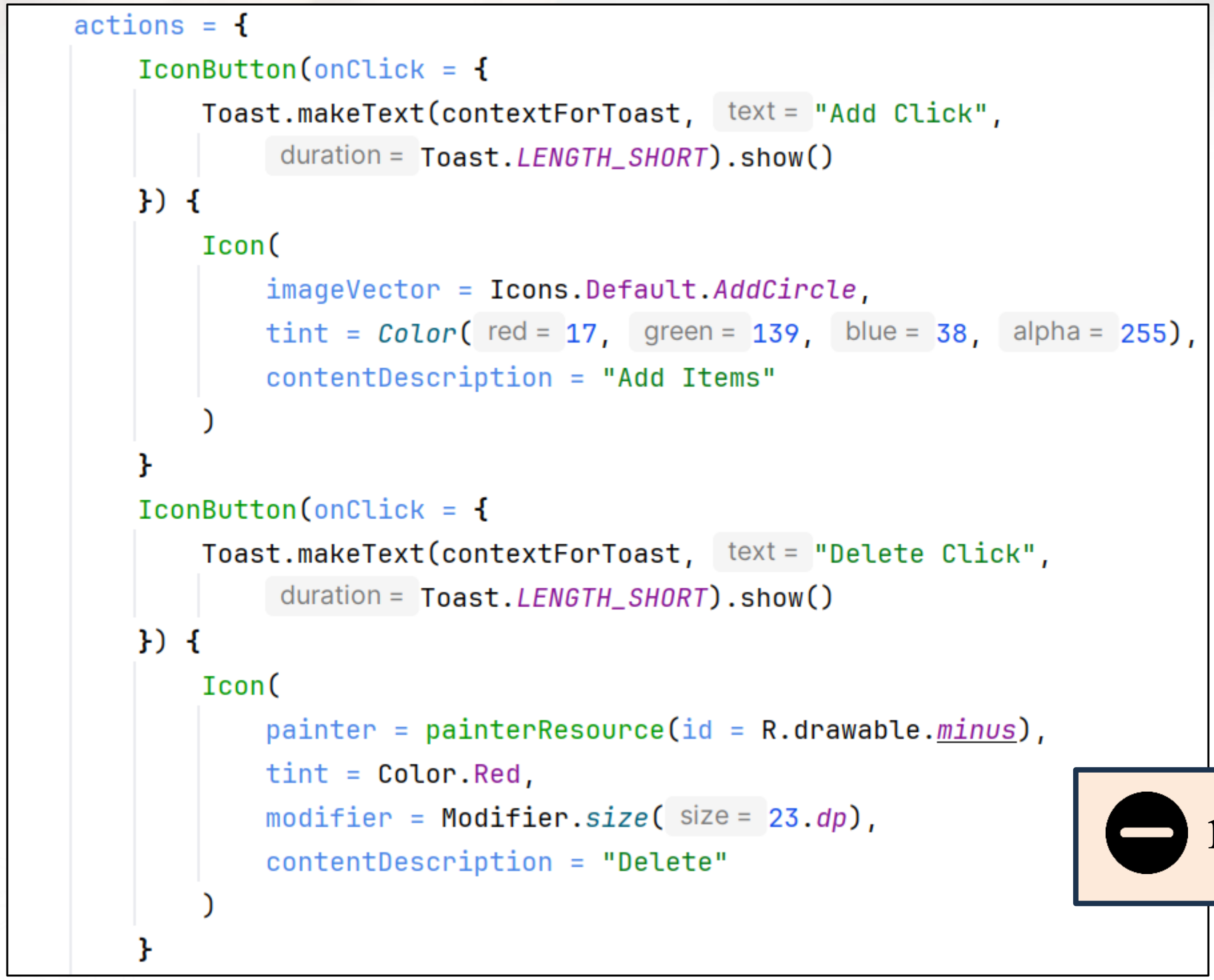
```
@OptIn( ...markerClass = ExperimentalMaterial3Api::class)
@Composable
fun MyTopBar(contextForToast: Context, onMenuClick: () -> Unit) {
    var expanded by remember { mutableStateOf( value = false) }
    CenterAlignedTopAppBar(
        title = {
            Text(text = "My Application", color = Color.DarkGray)
        },
        colors = TopAppBarDefaults.centerAlignedTopAppBarColors(
            containerColor = Color.Blue.copy(alpha = 0.1f)
        ),
        navigationIcon = {
            IconButton(onClick = {
                onMenuClick() // Call open Drawer function
            }) {
                Icon(imageVector = Icons.Default.Menu, contentDescription = "Menu")
            }
        },
```



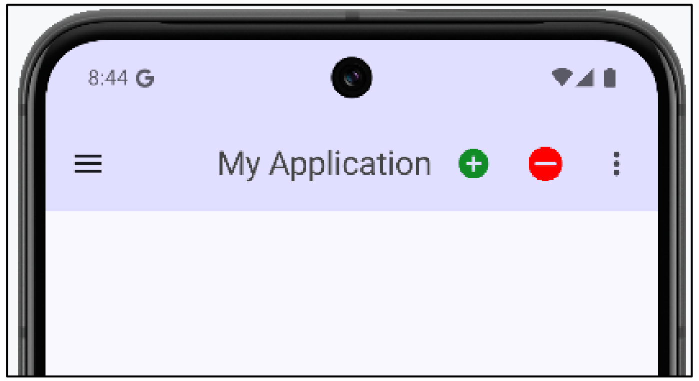


Scaffold & Navigation Components

MyTopBar Function(Cont.)



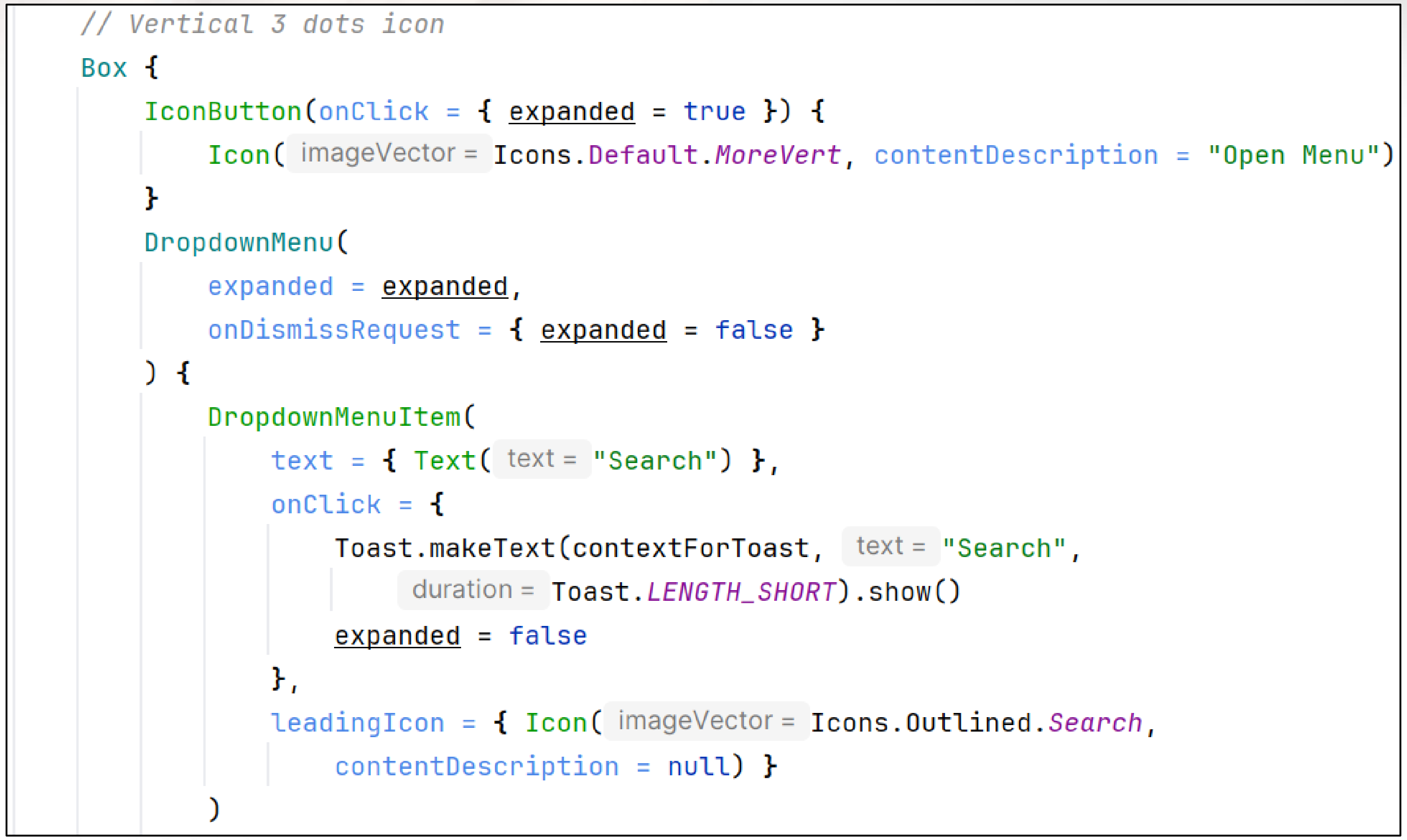
```
actions = {
    IconButton(onClick = {
        Toast.makeText(contextForToast, text = "Add Click",
            duration = Toast.LENGTH_SHORT).show()
    }) {
        Icon(
            imageVector = Icons.Default.AddCircle,
            tint = Color( red = 17, green = 139, blue = 38, alpha = 255),
            contentDescription = "Add Items"
        )
    }
    IconButton(onClick = {
        Toast.makeText(contextForToast, text = "Delete Click",
            duration = Toast.LENGTH_SHORT).show()
    }) {
        Icon(
            painter = painterResource(id = R.drawable.minus),
            tint = Color.Red,
            modifier = Modifier.size( size = 23.dp),
            contentDescription = "Delete"
        )
    }
```



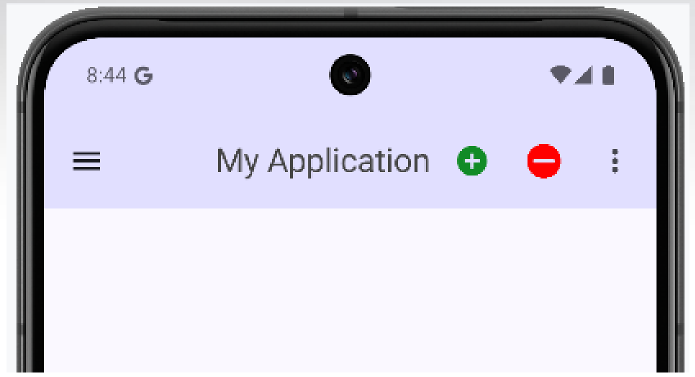
minus.png

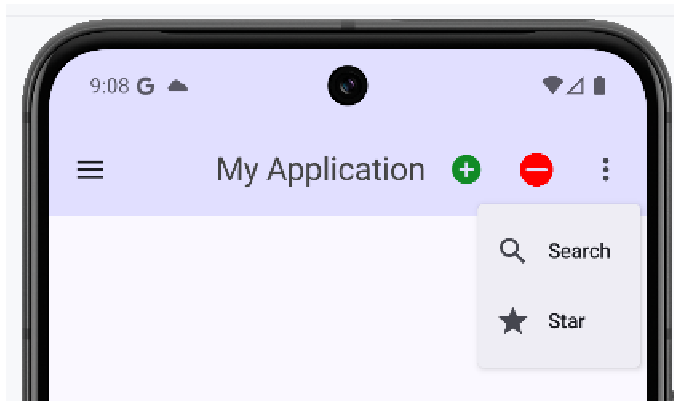
Scaffold & Navigation Components

MyTopBar Function(Cont.)



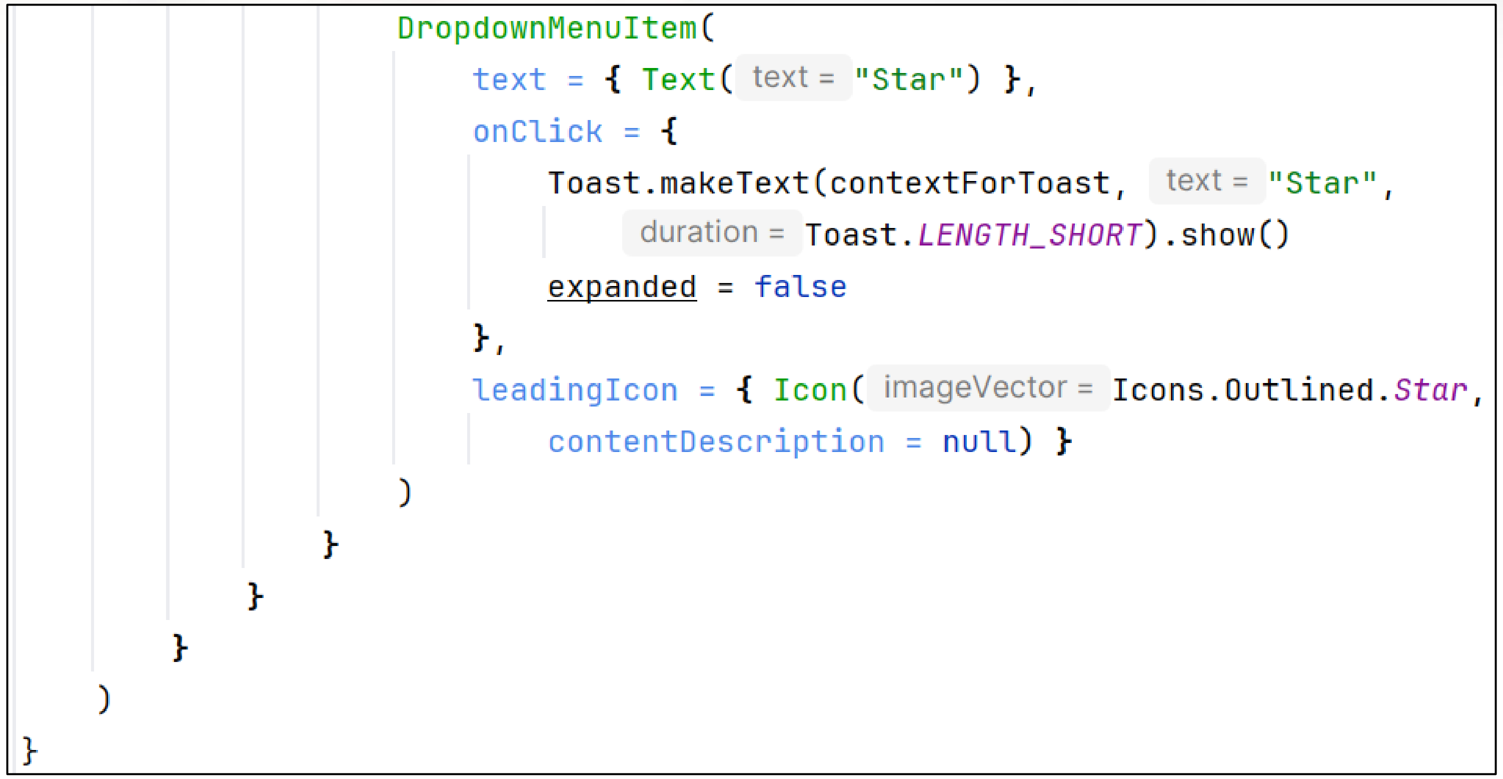
```
// Vertical 3 dots icon
Box {
    IconButton(onClick = { expanded = true }) {
        Icon( imageVector = Icons.Default.MoreVert, contentDescription = "Open Menu")
    }
    DropdownMenu(
        expanded = expanded,
        onDismissRequest = { expanded = false }
    ) {
        DropdownMenuItem(
            text = { Text( text = "Search") },
            onClick = {
                Toast.makeText(contextForToast,  text = "Search",
                     duration = Toast.LENGTH_SHORT).show()
                expanded = false
            },
            leadingIcon = { Icon( imageVector = Icons.Outlined.Search,
                contentDescription = null) }
        )
```



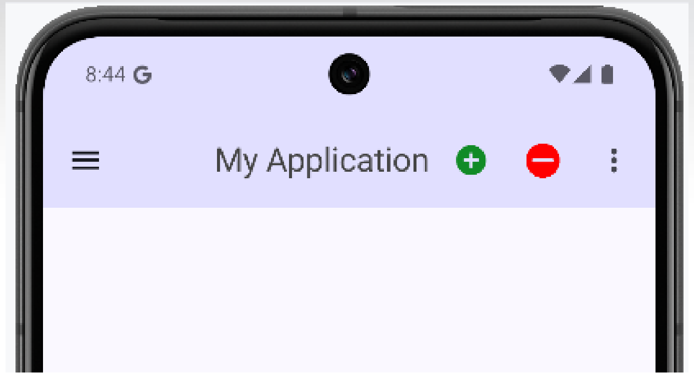


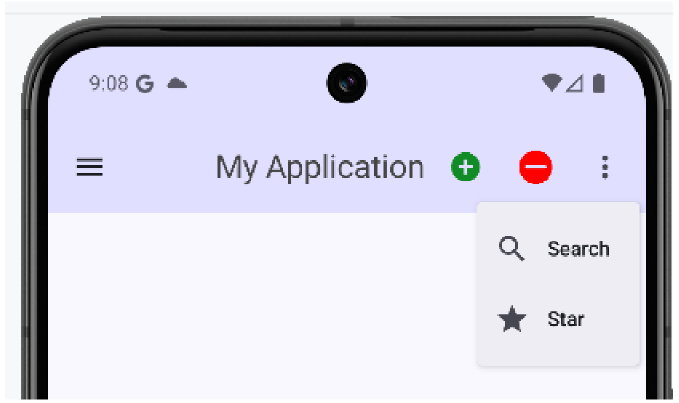
Scaffold & Navigation Components

MyTopBar Function(Cont.)



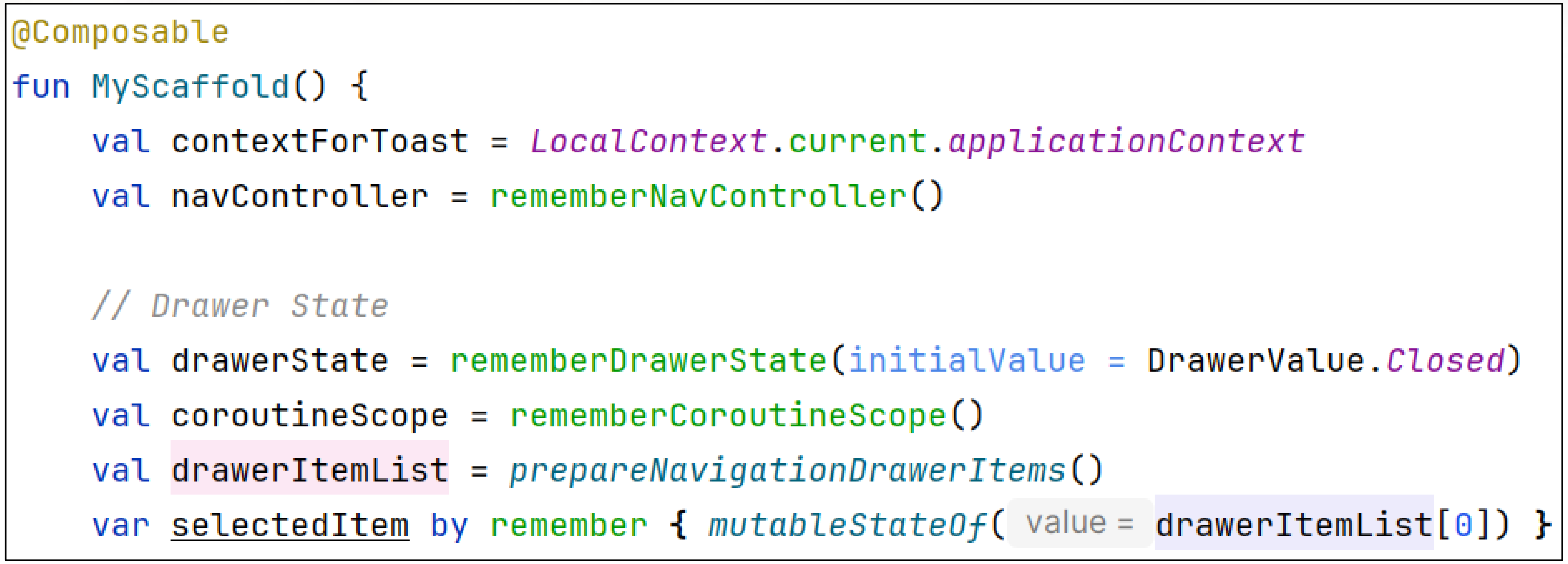
```
                    DropdownMenuItem(
                        text = { Text( text = "Star") },
                        onClick = {
                            Toast.makeText(contextForToast,  text = "Star",
                                duration = Toast.LENGTH_SHORT).show()
                            expanded = false
                        },
                        leadingIcon = { Icon( imageVector = Icons.Outlined.Star,
                            contentDescription = null) }
                    )
                }
            }
        }
    )
}
```



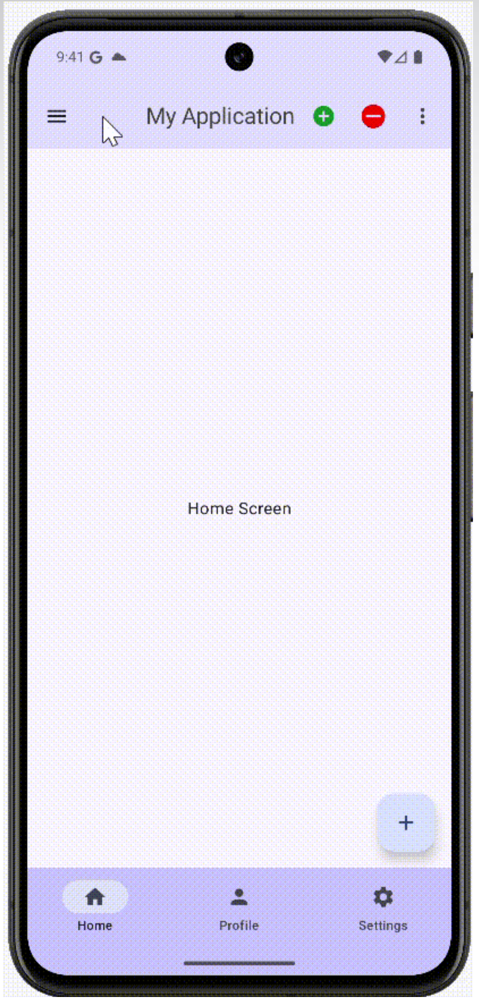


Scaffold & Navigation Components

MyScaffold Function

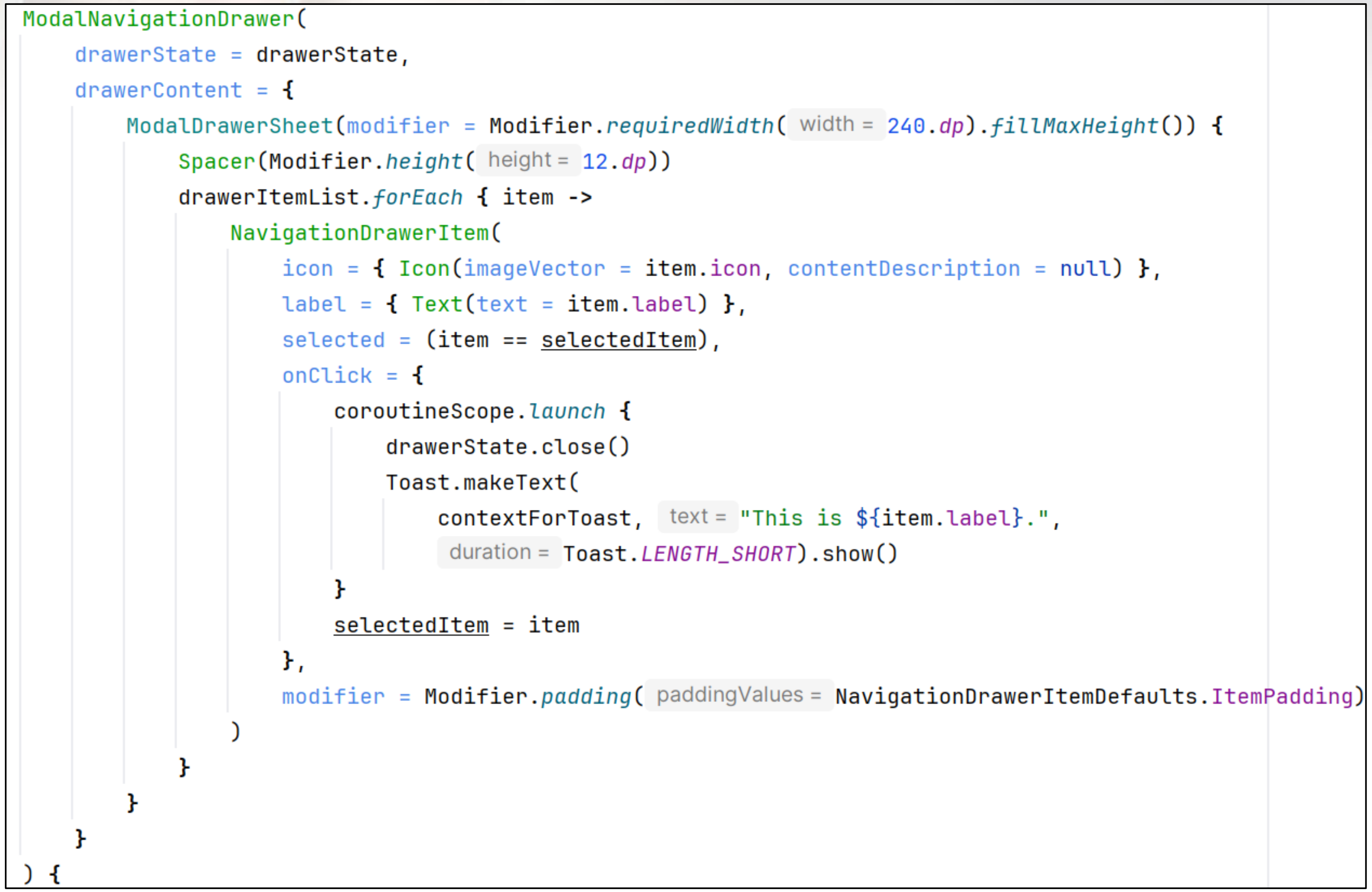


```
@Composable
fun MyScaffold() {
    val contextForToast = LocalContext.current.applicationContext
    val navController = rememberNavController()

    // Drawer State
    val drawerState = rememberDrawerState(initialValue = DrawerValue.Closed)
    val coroutineScope = rememberCoroutineScope()
    val drawerItemList = prepareNavigationDrawerItems()
    var selectedItem by remember { mutableStateOf( value = drawerItemList[0]) }
```

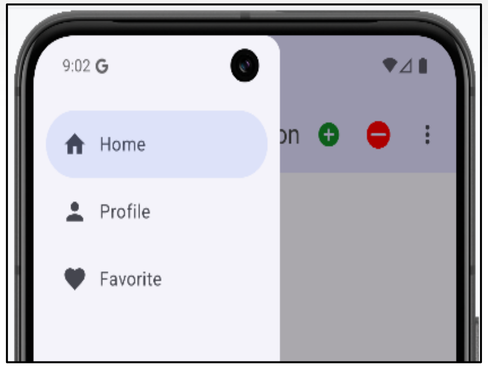


Scaffold & Navigation Components

MyScaffold Function (Cont.)

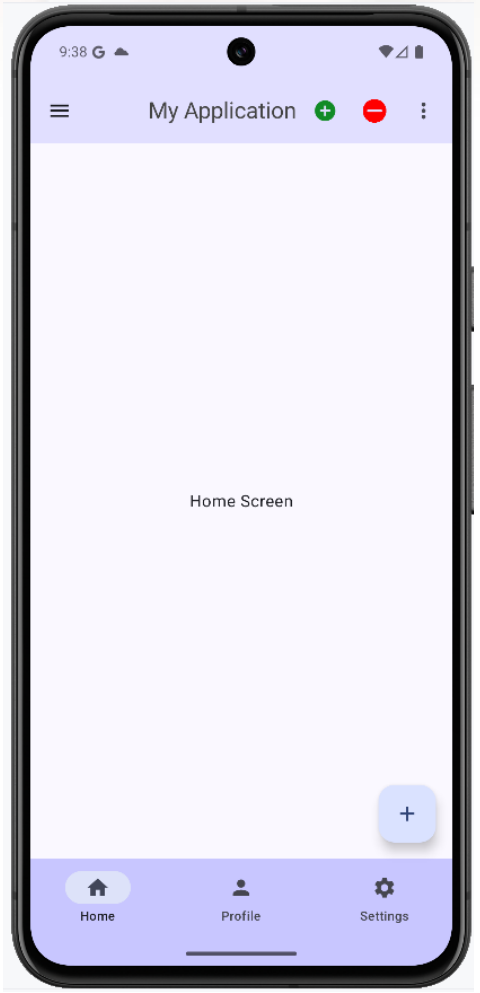


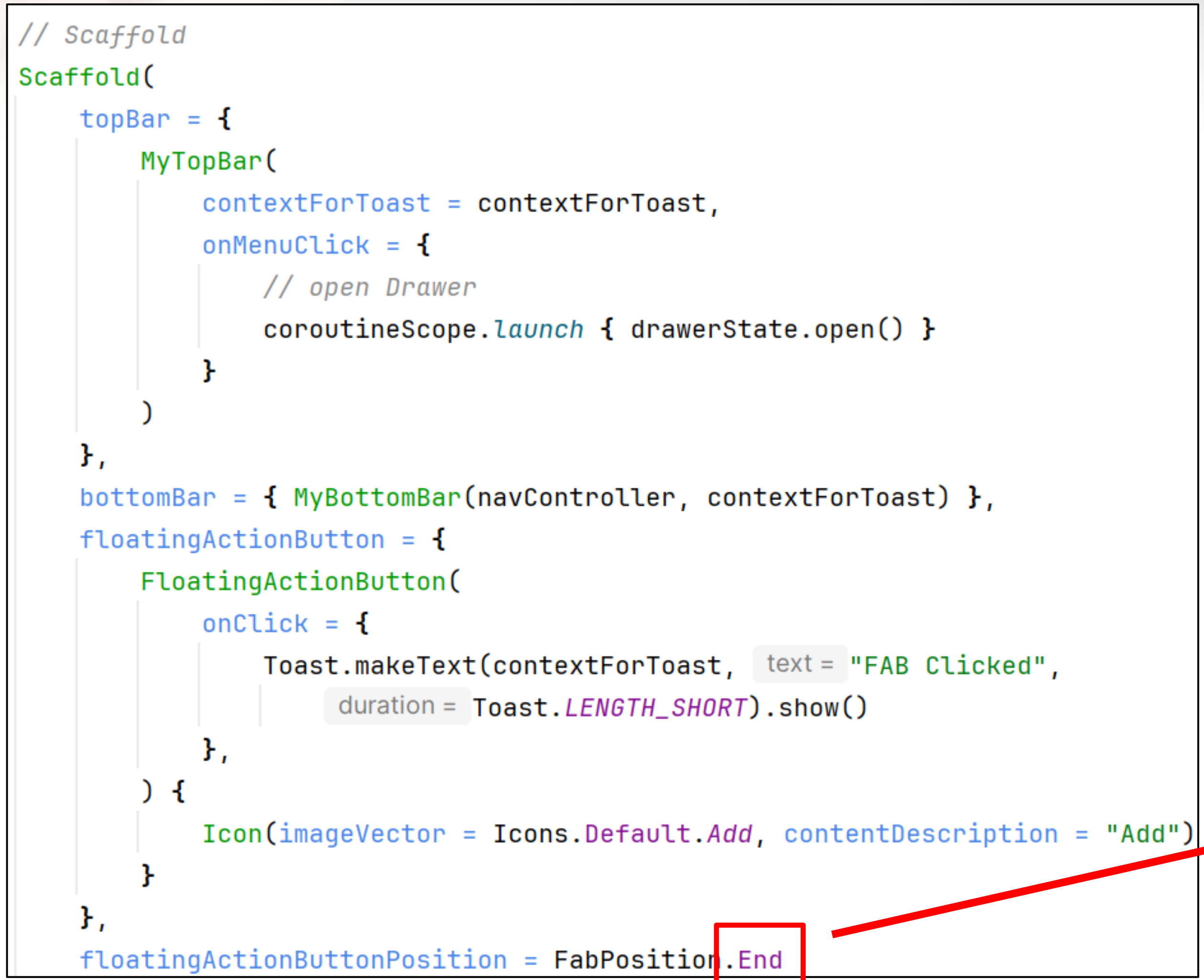
```
ModalNavigationDrawer(
    drawerState = drawerState,
    drawerContent = {
        ModalDrawerSheet(modifier = Modifier.requiredWidth( width = 240.dp).fillMaxHeight()) {
            Spacer(Modifier.height( height = 12.dp))
            drawerItemList.forEach { item ->
                NavigationDrawerItem(
                    icon = { Icon(imageVector = item.icon, contentDescription = null) },
                    label = { Text(text = item.label) },
                    selected = (item == selectedItem),
                    onClick = {
                        coroutineScope.launch {
                            drawerState.close()
                            Toast.makeText(
                                contextForToast,  text = "This is ${item.label}.",
                                 duration = Toast.LENGTH_SHORT).show()
                        }
                        selectedItem = item
                    },
                    modifier = Modifier.padding( paddingValues = NavigationDrawerItemDefaults.ItemPadding)
                )
            }
        }
    }
) {
```



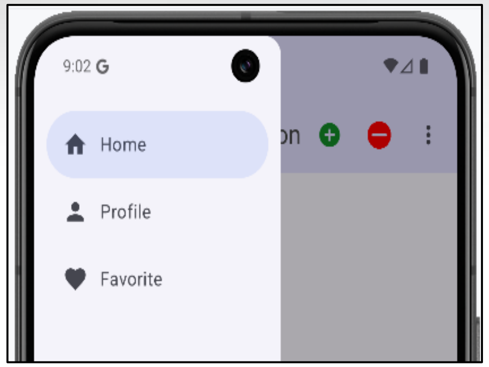
Scaffold & Navigation Components

MyScaffold Function (Cont.)





```
// Scaffold
Scaffold(
    topBar = {
        MyTopBar(
            contextForToast = contextForToast,
            onMenuClick = {
                // open Drawer
                coroutineScope.launch { drawerState.open() }
            }
        )
    },
    bottomBar = { MyBottomBar(navController, contextForToast) },
    floatingActionButton = {
        FloatingActionButton(
            onClick = {
                Toast.makeText(contextForToast, text = "FAB Clicked",
                    duration = Toast.LENGTH_SHORT).show()
            },
        ) {
            Icon(imageVector = Icons.Default.Add, contentDescription = "Add")
        }
    },
    floatingActionButtonPosition = FabPosition.End
```

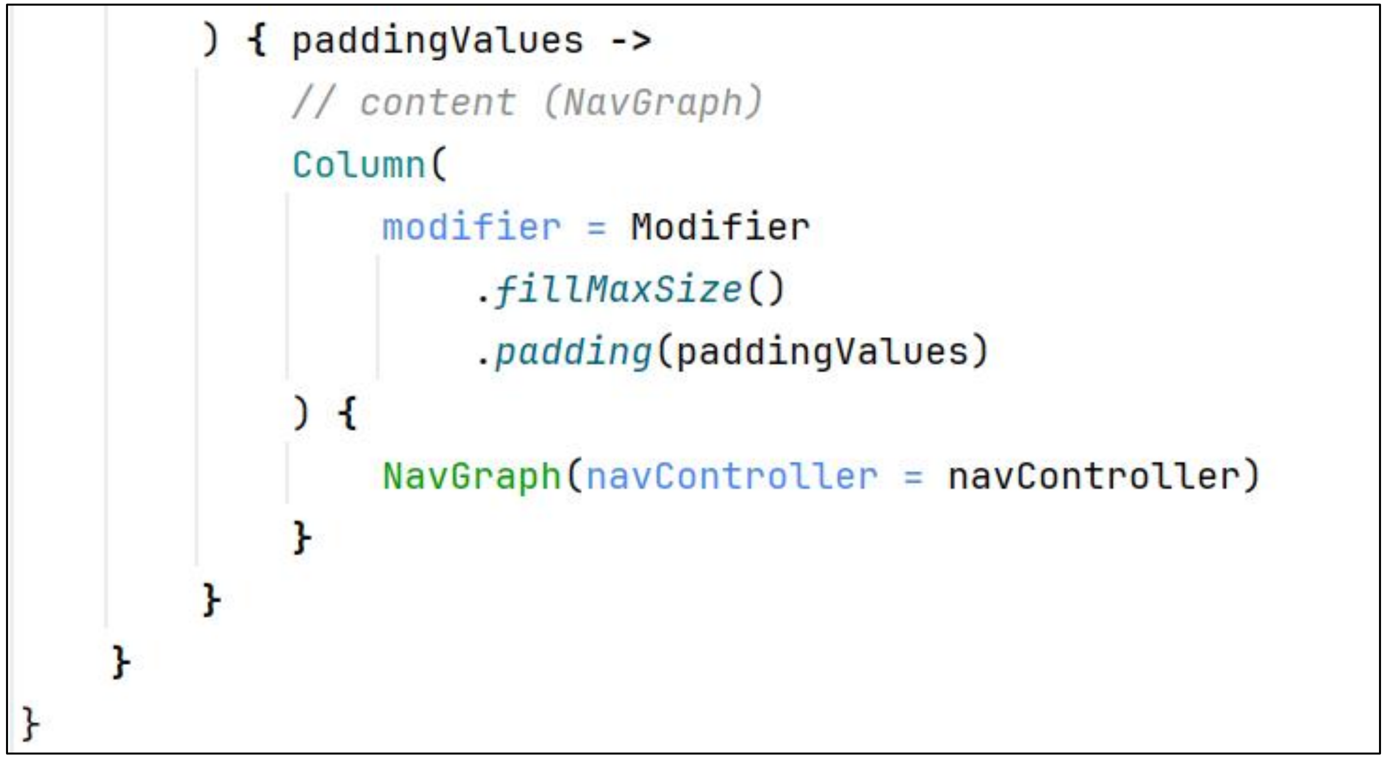


There are 4 positions :

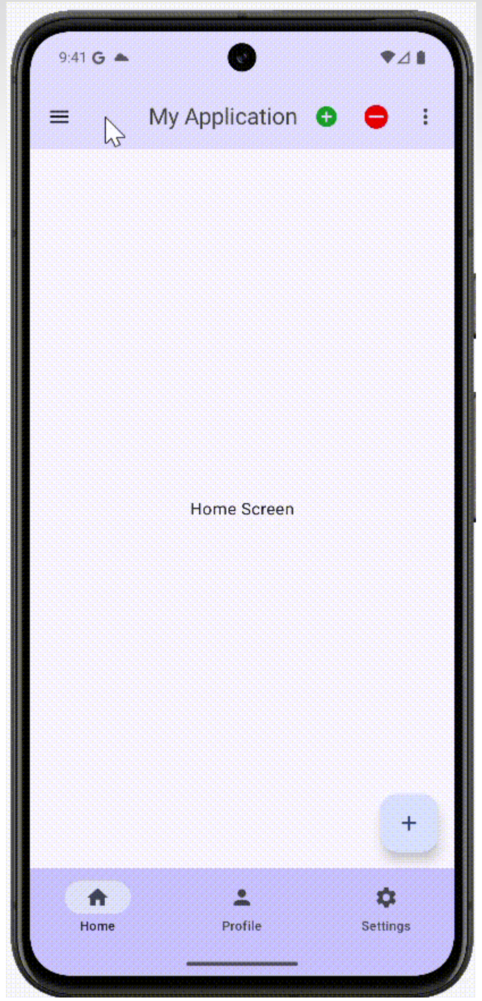
- FabPosition.Start
- FabPosition.Center
- FabPosition.End
- FabPosition.EndOverlay

Scaffold & Navigation Components

MyScaffold Function (Cont.)

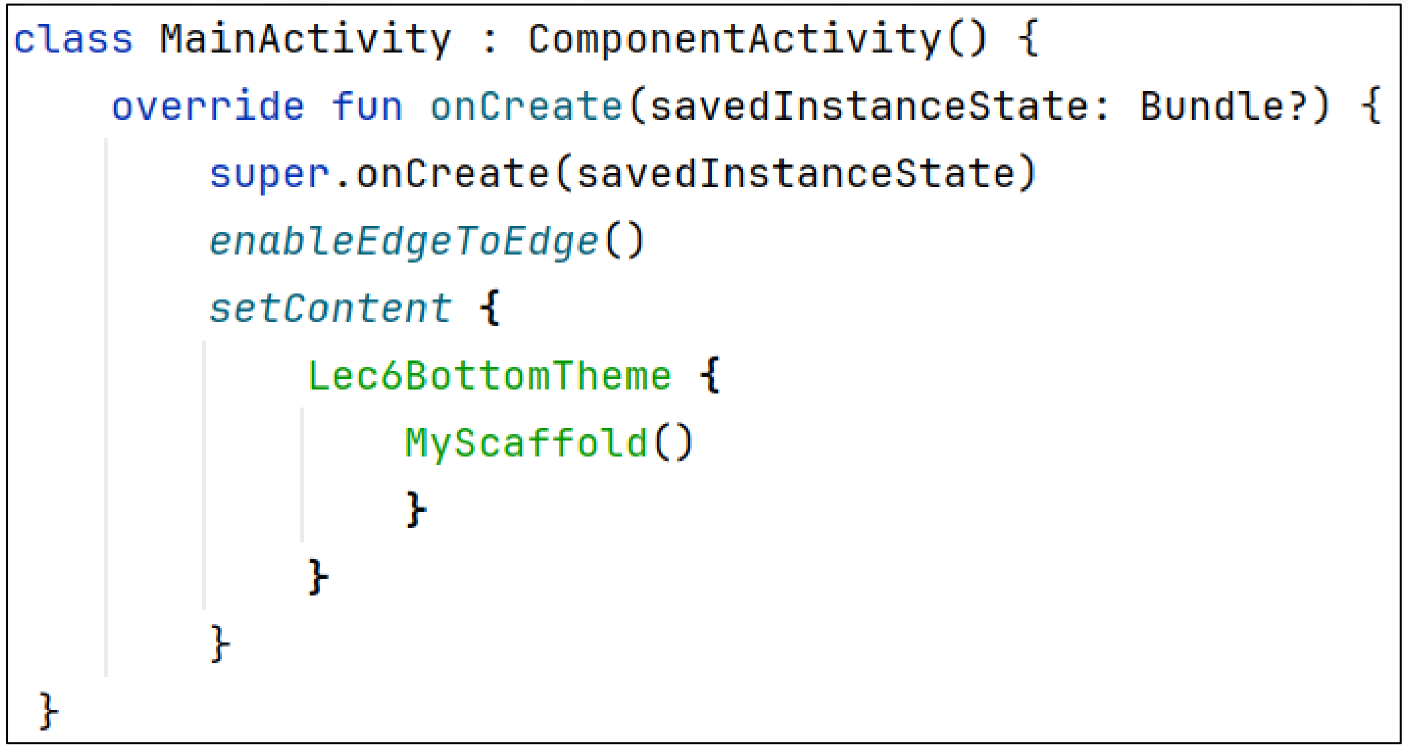


```
        ) { paddingValues ->
            // content (NavGraph)
            Column(
                modifier = Modifier
                    .fillMaxSize()
                    .padding(paddingValues)
            ) {
                NavGraph(navController = navController)
            }
        }
    }
}
```

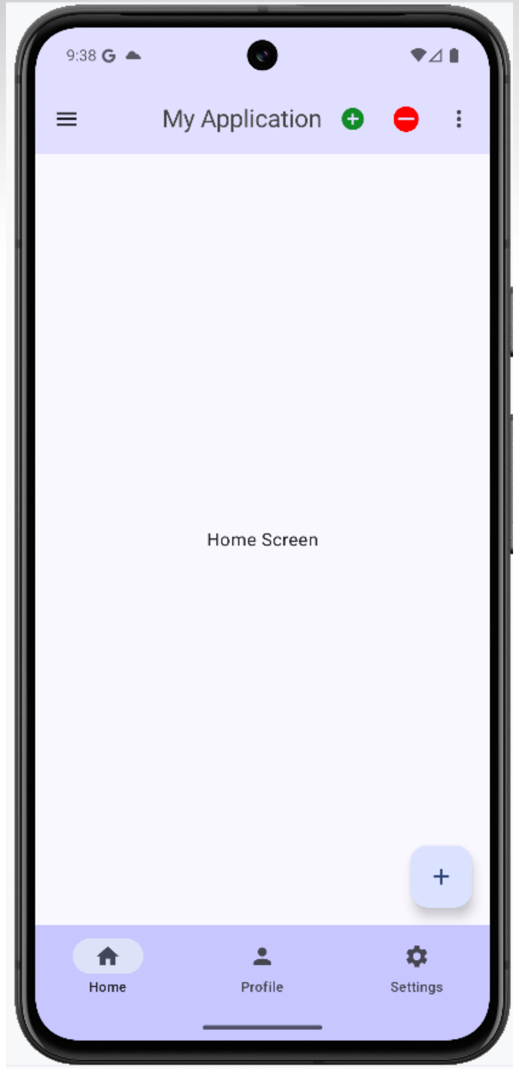


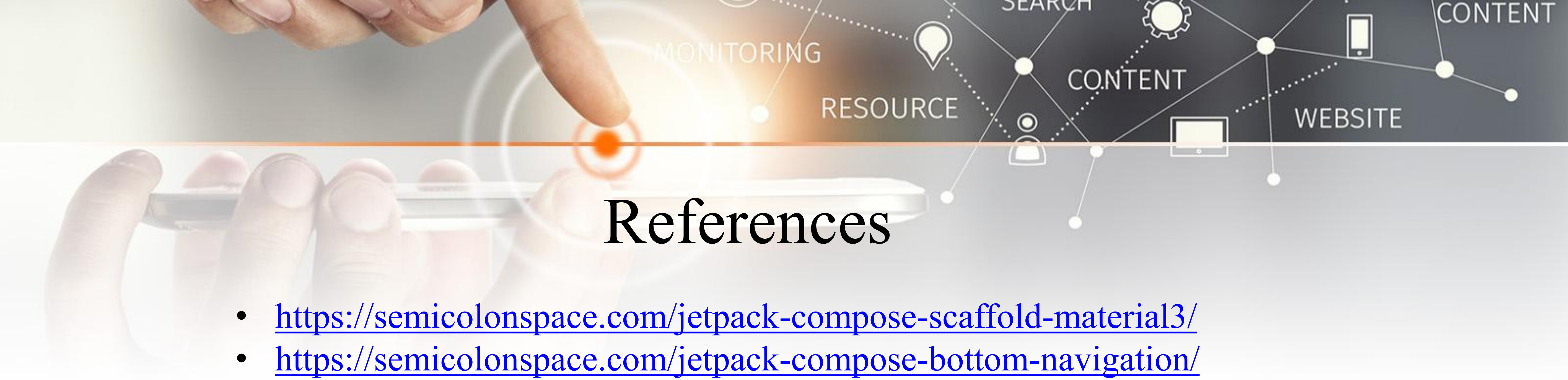
Scaffold & Navigation Components

class MainActivity (Cont.)



```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            Lec6BottomTheme {
                MyScaffold()
            }
        }
    }
}
```





References

- https://semicolonspace.com/jetpack-compose-scaffold-material3/
- https://semicolonspace.com/jetpack-compose-bottom-navigation/
- https://semicolonspace.com/jetpack-compose-navigation-drawer-material3/
- https://semicolonspace.com/jetpack-compose-dropdown-menu-material3/