BCS 371 Mobile Application Development I

Arthur Hoskey, Ph.D.
Farmingdale State College
Computer Systems Department

Today's Lecture

- NavigationBar
- Note: These slides assume that the Android Studio project comes with a Scaffold in the code that is given to you automatically.

- The content on these slides was modified from the following links:

Bottom Navigation Bar With Badges - UX With Material3 (video)
https://www.youtube.com/watch?v=c8XP_Ee7iqY

Navigation Bar with Jetpack Compose
https://bootcamp.uxdesign.cc/navigation-bar-with-jetpack-compose-32b052824b7d

NavigationBar

- The NavigationBar allows the user to easily go to different screens in your app.
- These slides describe setting up a material 3 NavigationBar using Compose.

- The default dependencies come with some icons. For example: home, add, delete.
- If you want access to a wider variety of icons, you can include the material extended icons library (use Gradle dependency below).
- **Important! This dependency is very large.** If you are going to use it then you should modify the file use Proguard open-sourced Java class file shrinker, optimizer, obfuscator, and preverifier tool.
- Helpful link:

https://stackoverflow.com/questions/75127384/how-can-i-access-all-material-icons-in-android-studio-through-icon-in-jetpack-co

- Here is the dependency (ONLY USE IF YOU REALLY NEED IT!):

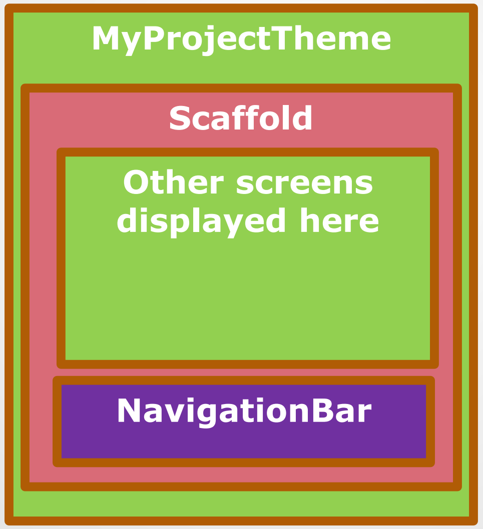
```
dependencies {
  ...
  var compose_version="1.5.4"
  implementation("androidx.compose.material:material-icons-extended:$compose_version")
}
```

Extended Icons Dependency

Overview of NavigationBar Setup

- A Scaffold is a composable function that is used to organize a complex user interface.
- It has places to store standard pieces of a user interface (navigation bar, top app bar, floating action button etc…).
- The Empty Activity project in Android Studio Ladybug comes with a Scaffold already included.

For Android Studio Ladybug and later the setContent block already has a Theme composable function generated for you (this will have a Scaffold inside of it). If the project is named MyProject then there will be a composable named MyProjectTheme which has the Scaffold inside of it.



Other screens are displayed inside the Scaffold. When the user navigates between screens this is the area that gets changed.

You can put a navigation bar in the Scaffold that is given to you. It will appear at the bottom of the window

Overview of Navigation Bar Setup

Overview of Navigation Bar Setup in the Composable

```
override fun MainActivity.onCreate(…) {
    // Other default code here…
    setContent {
        TestNavigationBarTheme {

            Scaffold (
                bottomBar = {
                    NavigationBar {
                        // Configure the NavigationBar here
                    }
                }
            ) {
              // Scaffold padding code AND NavHost/Screen display code
            }

        }
    }
}
```

MainActivity.onCreate contains the default Scaffold code

Use bottomBar of Scaffold to specify the NavigationBar. (this is a parameter to the Scaffold function)

The NavigationBar is a inside the Scaffold

Screens are displayed inside the Scaffold

Default code comes with a Scaffold in Android Studio Ladybut

Overview of Navigation Bar Setup in the Composable

Overview of NavigationBar Setup

1. Navigation Item Class. Create a class to store navigation item information (the navigation bar contains multiple nav items).
2. Create the NavigationBar (assumes Scaffold is given in default project code in MainActivity.onCreate)
   A. Create a list of the navigation item class (mentioned above).
   B. Declare helper variables in the composable.
   C. Add NavigationBar to the Scaffold.
   D. Configure the NavigationBar.

Overview of Navigation Bar Setup

Create Class to Store Navigation Item Information

- This class will have members to store the icon title and the icon resource.
- For example:

Material Design 3 suggests displaying different icons depending on if an item is selected or not

```
data class MyNavItem (
    val title: String,
    val iconSelected: ImageVector,
    val iconUnselected: ImageVector,
    val route: String
)
{
    // Variables already defined in the primary constructor
}
```

Define icons for both selected and unselected

route member. Only need to include the route member if using the NavigationBar in conjunction with a NavHost.

1. Create Class to Store Navigation Item Information

2A. Create a Navigation Bar Using Compose

Create a List of Navigation Items

- Put the navigation item list in the composable where the Navigation Bar is defined.
- The following code shows how to create a list of navigation items:

```
val navItemsList = listOf(
  MyNavItem(
    title="Home",
    iconSelected = Icons.Filled.Home,
    iconUnselected = Icons.Outlined.Home,
    route=""
  ),
  MyNavItem(title="Add",
    iconSelected = Icons.Filled.Add,
    iconUnselected = Icons.Outlined.Add,
    route=""
  ),
)
```

Material Design 3 suggests using a filled icon for selected items and an outlined icon for unselected items

The route member is being set to "" because it is not being used in this example. If a NavHost were being used, it should be set to a route for a given destination in the NavHost.

Important. Fill the route with a real value if using a NavHost

Declare Helper Variables in the Composable

- Declare a variable in the composable where the NavigationBar is defined to store the selected index of the NavigationBar.
- Declare other variables to get the destination.
- Here is the declaration:

RememberSaveable keeps the variable's value saved through configuration changes as well as recompositions

```
var selectedItemIndex by rememberSaveable { mutableStateOf(0) }
val navHostController = rememberNavController()

val navBackStackEntry by navHostController.currentBackStackEntryAsState()
val currentDestination = navBackStackEntry?.destination
```

Declare variable to hold the current destination (the current screen being displayed). The navBackStackEntry variable is needed to get the current destination.

2B. Declare Helper Variables in the Composable

2C. Add NavigationBar to the Scaffold

Add NavigationBar to the Scaffold

- The NavigationBar should go in the bottomBar of the Scaffold.
- For example:

bottomBar is a parameter for the Scaffold composable function

```
Scaffold (
    bottomBar = {

        NavigationBar {
            // Code to configure the NavigationBar goes here (see next slide)
        }


    }
)
{
   // Scaffold padding code and call to NavHost or other screen function goes here
}
```

Configure the NavigationBar

```
NavigationBar {
    navItemsList.forEachIndexed { index, item ->
        NavigationBarItem(
            selected = currentDestination?.hierarchy?.any { it.route.equals(item.route) } == true,
            onClick = {
                selectedItemIndex = index
                navHostController.navigate(item.route) {
                    popUpTo(navHostController.graph.findStartDestination().id) { saveState = true }
                    launchSingleTop = true
                    restoreState = true
                }
            },
            label = { Text(text = item.title) },
            icon = { Icon(contentDescription = item.title,
                    imageVector = if (index == selectedItemIndex) item.iconSelected
                        else item.iconUnselected
                )
            }
        ) // end – NavigationBarItem
    } // end – forEachIndexed
} // end - NavigationBar
```

forEachIndexed applies the lambda to all items in the list

If the currentDestination is the same as the item being processed in the foreach then its selected

When a nav item is clicked set the selectedItemIndex Then navigate to the screen for that nav item (assumes a NavHost is being used)

If the current nav item is selected, then use the selected icon otherwise use the unselected icon (this code uses an if expression).

2D. Configure the NavigationBar

Showing/Hiding the NavigationBar

Showing/Hiding NavigationBar

- Add a Boolean variable that decides if it should be displayed.
- Surround bottomBar in an if statement. If the Boolean variable is true, then show the NavigationBar.

Boolean variable for showing/hiding the NavigationBar

```
var showNavigationBar by rememberSaveable { mutableStateOf(false) }

Scaffold (
bottomBar = {
        if (showNavigationBar) {
            NavigationBar {
                // NavigationBar goes here
            }
        } // end - if (showNavigationBar)
    } // end - bottomBar
)
{ // Scaffold padding code and NavHost go here}
```

Only set the bottomBar if the Boolean variable is true (bottomBar is inside the if)

Changing Boolean Variable Value

Add code in other places in this function that will change showNavigationBar's value. For example, before navigating to another screen in the NavHost, set showNavigationBar to true if you want to show the NavigationBar.

- Now on to using a NavHost with the NavigationBar…

Using a NavHost with a NavigationBar

Overview of Using a NavHost with a Navigation Bar

- NavHost can be used with the NavigationBar.
- Make sure to include route data in the route member of the MyNavItem class.
- Composable containing Scaffold (inside MainActivity.onCreate in this example)
  - Declare local NavHostController variable inside setContent of MainActivity.onCreate (use rememberNavController).
  - The parameters in Scaffold ( ) are used to setup the Navigation Bar (bottomBar for the NavigationBar).
  - The body in Scaffold { } should call the NavHost composable function. Pass in the NavHostController as a parameter to the NavHost composable.
- Use the NavHost to navigate to different screens
  - Add code to the NavigationBarItem onClick that uses the NavHost to go to a specific destination (call NavController.navigate).
  - Use the following in the navigate function call block:

```
navHostController.navigate(item.route) {
    popUpTo(navHostController.graph.findStartDestination().id) {
        saveState = true
    }
    launchSingleTop = true
    restoreState = true
}
```

Avoid multiple copies of the same destination when reselecting the same item ← launchSingleTop = true

Restore state when reselecting a previously selected item ← restoreState = true

Using a NavHost with a NavigationBar

Add Navigation Code to NavigationBarItem onClick

- Add code to NavigationBarItem.onClick to navigate to the screen associated with the clicked item.

```
NavigationBar {
    navItemsList.forEachIndexed { index, item ->
        NavigationBarItem(
            // other code here
            onClick = {
                selectedItemIndex = index
                navHostController.navigate(item.route) {
                    popUpTo(navHostController.graph.findStartDestination().id) {
                        saveState = true
                    }
                    launchSingleTop = true
                    restoreState = true
                }
            },
        )
    }
}
```

item is an instance of MyNavItem

route is a member of MyNavItem and contains the route associated with the screen to navigate to

Use the navHostController to navigate to the screen associated with the navigation bar item that was clicked (the navHostController should be declared at the top of setContent block in MainActivity.onCreate

Using a NavHost with a NavigationBar

End of Slides

- End of Slides