CMPS 312 Mobile Application Development
Lab 7 - Navigation

In this lab, you will practice:

- Using Navigation to move between the app screens with Jetpack Compose
- Add Navigation UI components such as TopAppBar, Navigation Rail and BottomBar
- Integrating a custom tab bar composable into your navigation hierarchy
- Navigating with arguments
- Navigating using deep links

Part A – Jetpack Compose Navigation code lab

Complete the Jetpack Compose Navigation code lab.

Part B – Banking App

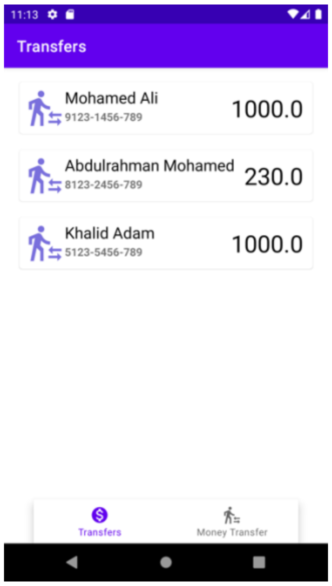
In this part, you will create a banking app where you will navigate and pass data between screens.

1. Create a new project under the **Lab7\BankingApp** folder on your GitHub repo.
   - Name the project BankingApp.
   - Use cmps312.bankingapp as the package.
   - Select *Empty Compose Activity* as the project template.

   Add the following dependency in the module's build.gradle:

   implementation "androidx.navigation:navigation-compose:{latest_version}"

2. Implement the **Transfer List** and **Transfer Details** use cases as shown in Figure 1 and Figure 2. When the user selects from the ***Transfers*** menu option, the app should navigate to the *Transfers Screen*. Then, when the user clicks a transfer from the list then the app should navigate to the ***Transfer Details Screen*** as shown in Figure 2.

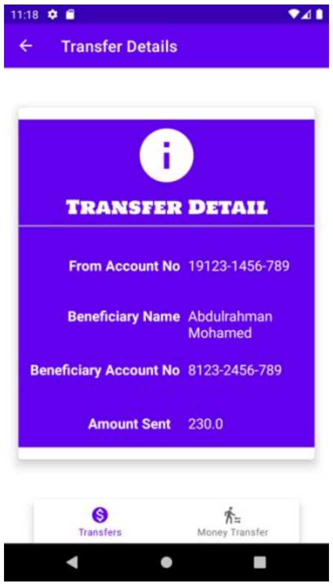

Transfers

Mohamed Ali 9123-1456-789 1000.0

Abdulrahman Mohamed 8123-2456-789 230.0

Khalid Adam 5123-5456-789 1000.0

Figure 1 Transfer List with Bottom Nav


Transfer Details

TRANSFER DETAIL

From Account No 19123-1456-789

Beneficiary Name Abdulrahman Mohamed

Beneficiary Account No 8123-2456-789

Amount Sent 230.0

Figure 2 Transfer Detail with Bottom Nav

a. Create a new kotlin file named BankingViewModel inside the viewmodel package.

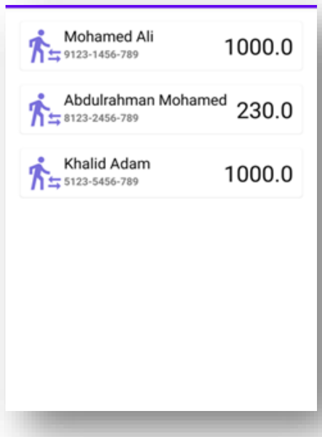
```
class BankingViewModel(appContext: Application) : AndroidViewModel(appContext) { ... }
```

The BankingViewModel should extend AndroidViewModel to get access to the appContext to be able to read json files from the assets folder.

b. Implement BankingViewModel class to hold the list of transfers.
Inside the class declare a transfers mutable state list, that will hold all the transfers. Any change that happens to this list will trigger UI recomposition

```
val transfers = mutableStateListOf<Transfer>( ... )
```

c. Open the TransferList Composable and display the list of transfers in a Lazy Column. The TransferList should get an instance of the BankingViewModel to access the transfers list.
Test your implementation, the app should display the screen shown in Figure 3.



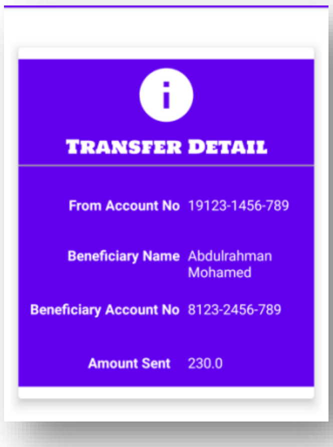
Mohamed Ali
9123-1456-789
1000.0

Abdulrahman Mohamed
8123-2456-789
230.0

Khalid Adam
5123-5456-789
1000.0

Figure 3. Transfer List screen

d. Implement the Transfer Details Screen composable. This composable should receive an Id.
- Get an instance of the BankingViewModel to getTransfer by transferId.
- Display the transfer details.



TRANSFER DETAIL

From Account No	19123-1456-789
Beneficiary Name	Abdulrahman Mohamed
Beneficiary Account No	8123-2456-789
Amount Sent	230.0

Figure 4. Detail List screen

e. Add a click listener to TransferList. Whenever, the user clicks on a transfer, the app should navigate them to the Transfer Details Screen and pass the selected **transferId** as a parameter.

f. Create a Sealed Class for Navigation Destinations. Each of the objects should include the title, icon and route

g. Setup the Navigation Host Composable and name it MyNavHost

h. Add a click listener to `TransferList`. Whenever, the user clicks on a transfer, the app should navigate them to the Transfer Details Screen and pass the selected `transferId` as a parameter.

i. Create a scaffold with bottom bar and top app bar. You should use the TopAppBar and NavigationBar composable. Do not create your own.

j. Once done the app should look like what you have seen in Figure 1 and 2 above.

3. In part B your task is to implement the *money transfer* use case. As shown in Figure 5, first the user selects the From Account and specifies the Transfer Amount. Then the user selects a beneficiary. Upon confirmation the transfer is added to the transfers list.

   a. In the FundTransfer screen get add the list of accounts, display them in the from account dropdown. When next is clicked

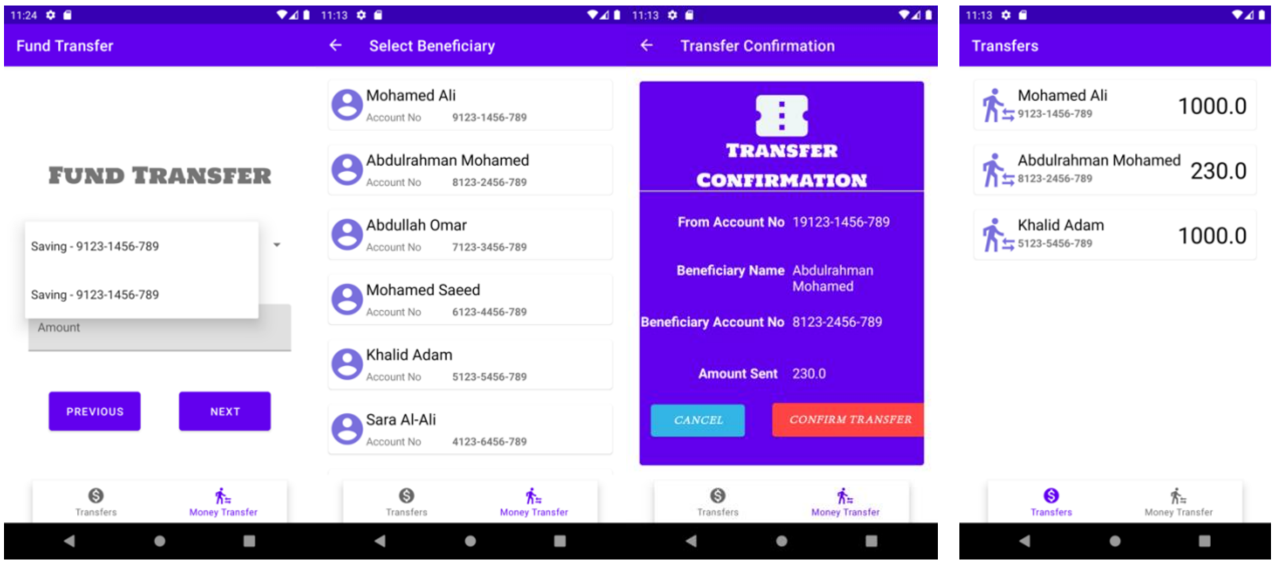
      i. assign the From Account and the Amount to bankingViewModel.newTransfer (

      ii. navigate to the Beneficiary screen.

   b. In the Beneficiary screen, display the list of beneficiaries to allow the user to select one of them. When next is clicked (a) assign the selected beneficiary details to bankingViewModel.newTransfer (b) navigate to the Confirmation screen.

   c. In the file Confirmation screen display the transfer details and allow the user to confirm the transfer.

   d. Finally, when confirm is clicked, add the new transfer object to the list of transfers then navigate to the Transfers List (tip: use popUpTo navigation options to clear the backstack up to and including the Transfers List).



Saving - 9123-1456-789
Saving - 9123-1456-789

Mohamed Ali	Account No 9123-1456-789
Abdulrahman Mohamed	Account No 8123-2456-789
Abdullah Omar	Account No 7123-3456-789
Mohamed Saeed	Account No 6123-4456-789
Khalid Adam	Account No 5123-5456-789
Sara Al-Ali	Account No 4123-6456-789

From Account No 19123-1456-789
Beneficiary Name Abdulrahman Mohamed
Beneficiary Account No 8123-2456-789
Amount Sent 230.0

Mohamed Ali	9123-1456-789	1000.0
Abdulrahman Mohamed	8123-2456-789	230.0
Khalid Adam	5123-5456-789	1000.0

Figure 5 Add Transfer