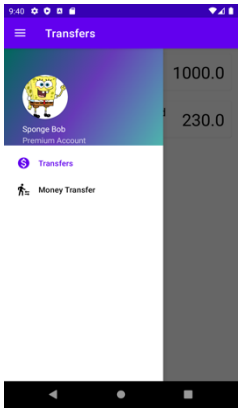
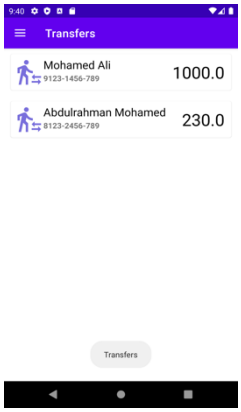
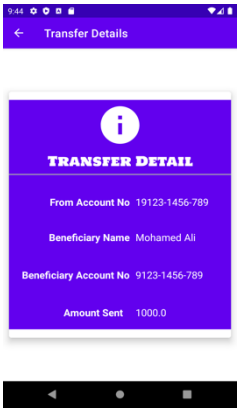
CMPS 312 Mobile Application Development
Lab 7 – Model-View-ViewModel (MVVM) Architecture

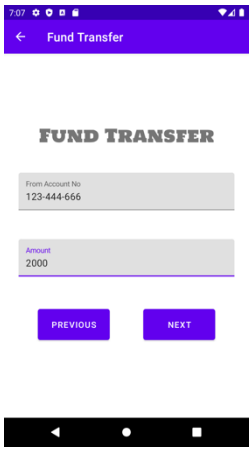
Objective

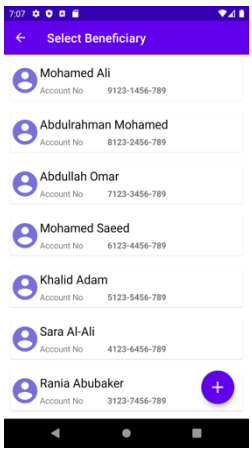
In this Lab, you will **continue building the Banking App** and practice MVVM architectural design pattern. In particular, you will practice using ViewModel, LiveData and Data Binding to LiveData objects.

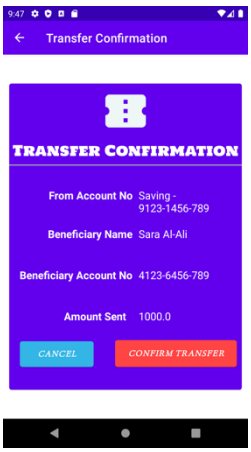


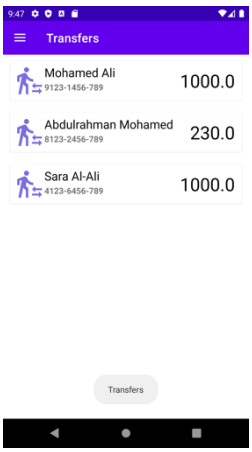










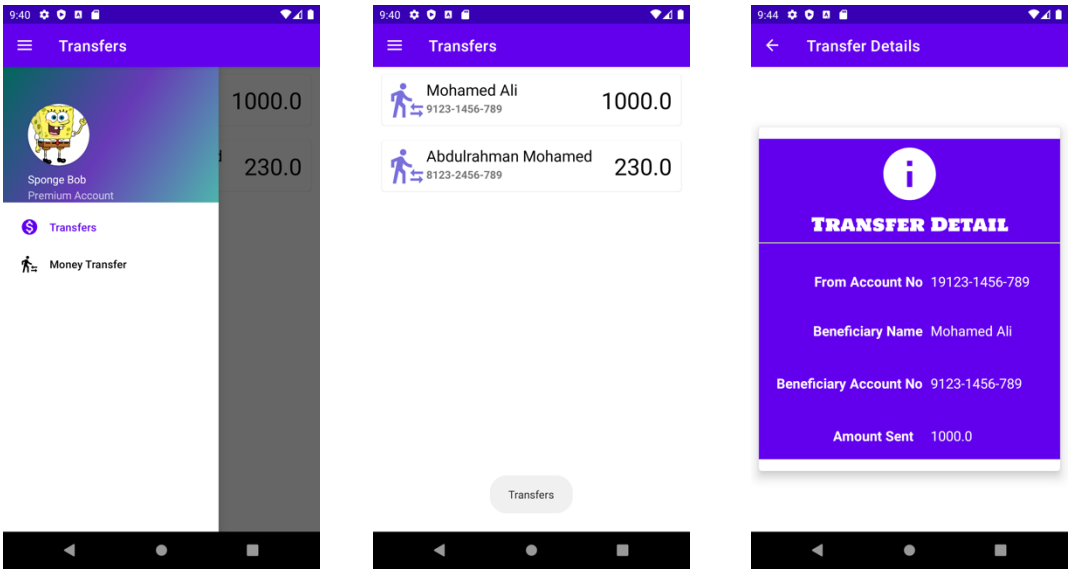


Preparation

1. Sync the Lab GitHub repo and copy the **Lab7-MVVM** folder into your repository.
2. Open the project ***Bank App*** in Android Studio. The project has the following folders and files:
    - **layout**: all the layouts for the fragments and adapter.
    - **adapters**: two adapters. One for the beneficiaries list and one for transfers list
    - **model:** Account, Transfer and Beneficiary
    - **repository:** the repository class that reads from the accounts.json, transfers.json and beneficiaries.json is provided
    - **drawable:** images needed for this app.
    - **assets:** contains the beneficiary.json and transfers.json files
    - **navigation:** navigation Graph

PART A: Implementing the Transfer and Transfer Details

In Part A, your task is to implement the transfers and transfer details screens. When the user selects from the list of transfers, you should navigate them to the transfer details screen and show the details of the selected transfer.



1. Inside the transfer, package create a new package named **viewmodel**
2. Create a class called **TransferViewModel** that is going to hold the list of transfers
3. Inside the class create one mutable live data object called **_transfers**
   ```
   private var _transfers = MutableLiveData<MutableList<Transfer>>()
   ```
4. You should expose the _transfers mutable live data variable through a function called transfers().
   ```
   fun transfers(): LiveData<MutableList<Transfer>> = _transfers
   ```
5. Initialize the _transfers using the BankingRepository object. You should do the initialization inside the init function.
6. Open the **TransferListFragment** and create a **TransferViewModel** object named **transferViewModel**
7. Use this object to observe the **_transfer live data**. Once a change happens to the list of **_transfers** inside the **transferViewModel**, you should then populate that list inside the **TransferListAdapter**
8. Open the **graidle app module** and add the following line of code that enables the data binding feature for the app.
   ```
   buildFeatures {
       dataBinding true
   }
   ```
9. Open, **TransferListAdapter** and **list_item_transfer** layout file and implement the necessary data binding needed to display the list of transfers.
10. Handle the onclick on the list items. So, once the user clicks on a list item, then are navigated to the **TransferDetailsFragment.**
11. Open **TransferViewModel and** create one more object called **selectedTransfer**

12. Open the **showDetails** function inside the **TransferListFragment** and assign the passed transfer object to the **selectedTransfer** inside **TransferViewModel**
13. Do all the necessary data binding needed for the **TransferDetailsFragment** to display the selectedTransfer object inside the **TransferViewModel**

PART B: Implementing the Transfer Component

In PART B your task is to implement the transfer component using the same scenario that you have seen in Part A. This transfer component allows the user to transfer money to a given beneficiary. Once the user completes a particular transfer, that transfer is then added to the list of **transfers** that you have shown in part A. Refer to the images below to understand the requirement.

To implement this you will need the following.

1. You should implement the spinner drop down for the accounts so the user selects from a list
2. You need to create an object called **transfer** inside the **TransferViewModel** that is going to hold the transfer data that is needed for the three screens [Fund Transfer, Select Beneficiary, and Transfer Confirmation].
3. You should create a view model for the beneficiaries component
4. You should modify the **BeneficiaryAdapter** and add all the necessary binding to display the list of beneficiaries.
5. Finally, when the user presses on the confirm button, you should take the transfer object and add it to the _transfers list inside the **TransferViewModel**