CMPS 312 Mobile App Development
Banking App
Lab Assignment 3
Deadline - Thursday October 29, 2020

Objective

The objective of this assignment is to practice building an android app that communicates with a Web API using retrofit and coroutines. You will also practice using the navigation component, View Models, Databinding and Live Data.

Overview

In the assignment, you will design and implement a **Bank App** that allows the bank employees to add, update, delete accounts. A web version of the app is available at https://employee-bank-app.herokuapp.com. Your task is to implement a mobile app to allow listing, adding, updating and deleting accounts.

Implementation Instructions

You should write the whole app by yourself no base solution is provided. You can use the skills acquired in labs **6, 7, 8 and 9** to help you deliver the app.

1. Create a new project and name it "**Banking App**" under your repository and add all the needed dependencies. You can copy the last lab dependencies and they should be enough for you to complete this project.
2. Add one model class [**Account**] that can store the Account objects that you will get from the Web API. An example account object in JSON format is shown below.

```
{
    "accountNo": "1602821627042"
    "name": "Abdulahi",
    "acctType": "Saving",
    "balance": 10888,
}
```

3. Use the Retrofit library to communicate with the Web API. First, create a Service Interface and a Repository to get, add, update, and delete accounts using the Web APIs shown in the table below and available at https://employee-bank-app.herokuapp.com. You can **test the Web API using Postman** to better understand what each of the following URLs returns and how to call them.

HTTP Verb	Url	Functionality
Get	/api/accounts	Get all counts
Get	/api/accounts/:accountNo	Get an account by account no
Post	/api/accounts	Add an account
Put	/api/accounts/:accountNo	Update an account
Delete	/api/accounts/:accountNo	Deletes an account by account no

4. Create the app navigation graph, named **nav_graph**, to allow the user to navigate within the app to access **Accounts List**, **Add account, and Update account** (As shown in figure 1 and 2).
5. Design and implement the layout and the code for list, add, update and delete account. All of your layouts should use **data binning** whenever possible.
   - Implement account list fragment and show in Figure 1.
   - Implement the delete account. When the user clicks on the delete button, you should remove the account from both the list as well as from the server by calling the Web API.
   - Implement the add and update account scenarios as shown in fig. 1 and fig 2. You may use the same fragment for Add & Update account.

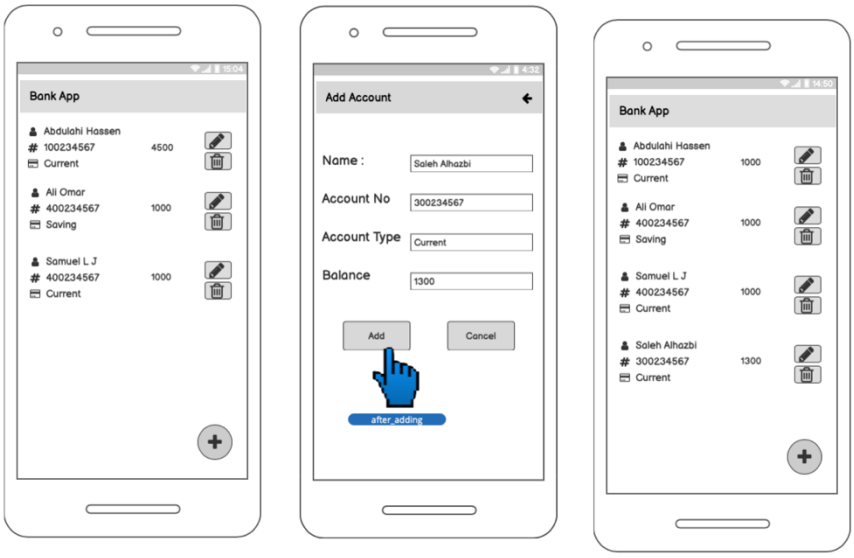

Figure 1. List and Add Account

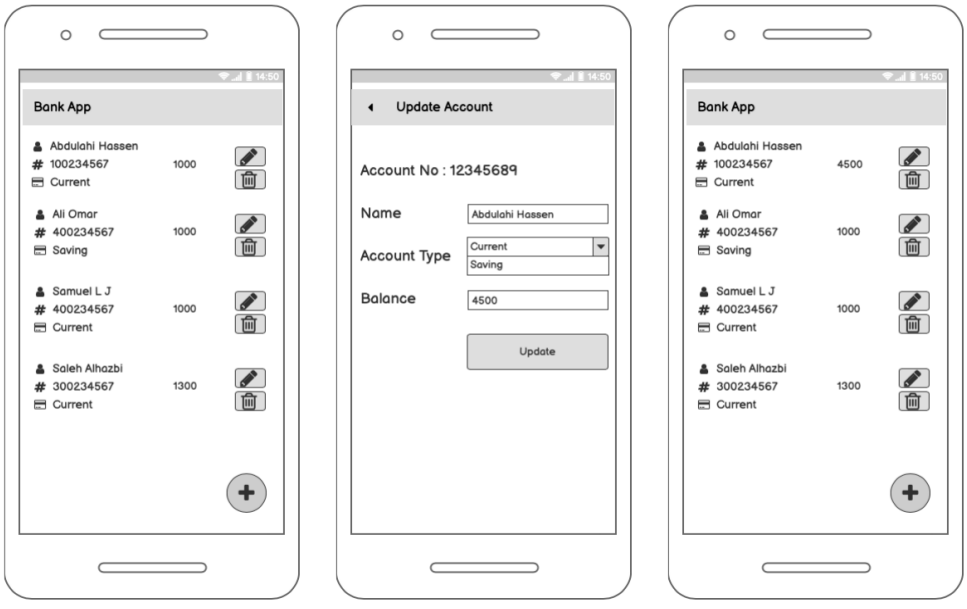

Figure 2 Update Account

Submit the testing sheet as well as the code under "**your_repository/assignments/assignment3**"

Grading Rubrics

Criteria	%	Functionality*	Quality of the implementation
Accounts List	40		
Delete account	15		
Add account	25		
Update account	20		
Total	**100**		
Provide screenshots in *testing.docx*	-		*[-10pts if missing]*

* **Possible grading for functionality**: ***Complete and Working*** (get 70% of the assigned grade), ***Complete and Not working*** (lose 60% of assigned grade) and ***Not done*** get 0. The remaining grade is assigned to the quality of the implementation. Must submit screenshots in **Testing-GradingSheet.docx** (otherwise -10pts).