Mobile App Development
In Class Assignment 04

Basic Instructions:

1. In every file submitted you MUST place the following comments:
    a. Assignment #.
    b. File Name.
    c. Full name of the student.
2. Each group is required to submit the assignment on Canvas.
3. Please download the support files which include a Java project to be used for this assignment.
4. **Submit Codes:**
    a. Zip all the project folder to be submitted on canvas.
5. Submission details:
    a. The file name is very important and should follow the following format: **Group#_InClass04.zip**
    b. You should submit the assignment through Canvas: Submit the zip file.
6. **Failure to follow the above instructions will result in point deductions.**

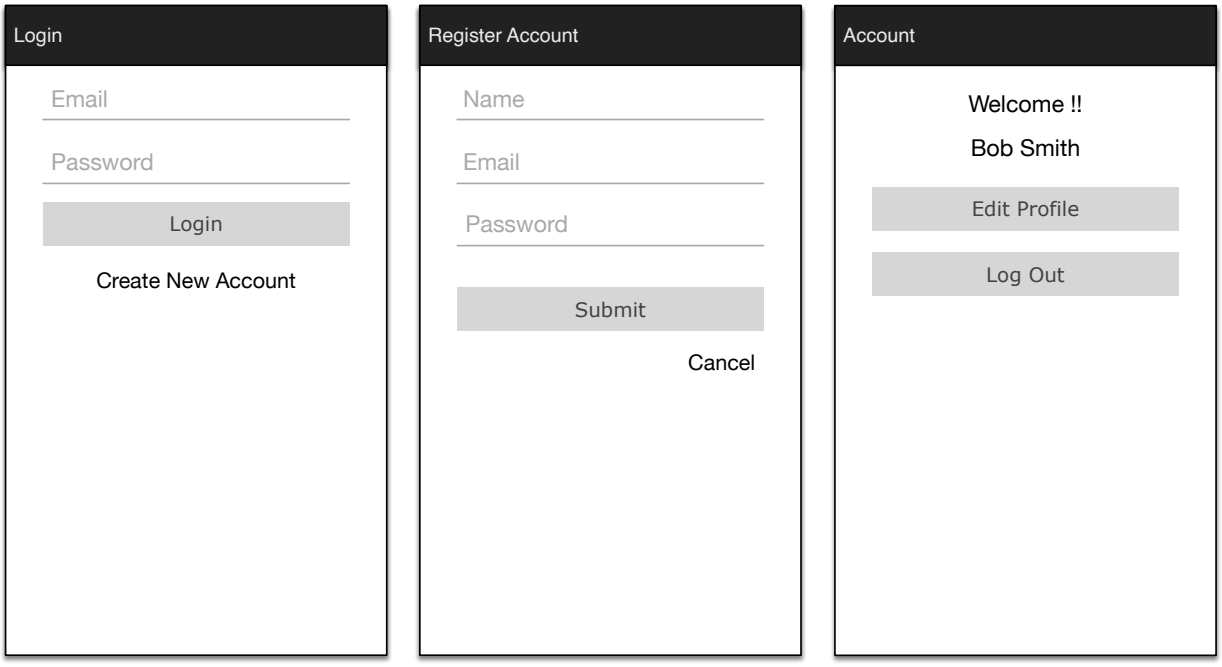
In Class 04 (100 Points)

In this assignment you will be building an application that uses as single activity and multiple fragments and exchanges data among these fragments. Structure and important setup are listed below:

- In this app you will have only one Activity and 4 fragments, all communication between activities should be managed by the activity.
- You are provided with a DataServices class file which you need to import in your project. Make sure to update the package name to match your project's.
  - The DataServices class will emulate the account management functions such as login, register, and update functions required to login, register and update accounts. All the provided functions return an AccountRequest object, below is an example code snippet showing how to use the login function.

```
DataServices.AccountRequestTask task = DataServices.login("a@a.com", "test123");
if(task.isSuccessful()){ //successful
    DataServices.Account account = task.getAccount();
} else { //not successful
    String error = task.getErrorMessage();
    Toast.makeText(this, error, Toast.LENGTH_SHORT).show();
}
```

Figure 1, Example Code Snippet



(a) Login Fragment (b) Register Fragment (c) Account Fragment

Figure 2, Application Wireframe

Part 1 : Login Fragment (20 Points)

The interface should be created to match the UI presented in Figure 2(a). The

requirements are as follows:

1. Upon entering the email and password, clicking the submit button should:
    a. Clicking “Login” button, if all the inputs are not empty, you should attempt to login the user by calling the DataServices.login function.
    b. If there is missing input, show a Toast indicating missing input.
    c. If the login is successful, then communicate the returned account with the activity and replace the current fragment with the Account Fragment.
    d. If login is not successful, show a Toast message indicating that the login was not successful.
2. Clicking the “Create New Account” should replace this fragment with the Register Fragment.

Part 2 Register Fragment (20 Points)

The interface should be created to match the UI presented in Figure 2(b). The requirements are as follows:

1. This fragment should allow a user to create a new account. Upon entering the name, email and password, clicking the Submit button should:
    a. If all the inputs are not empty, you should attempt to signup the user by calling the DataServices.register function.
    b. If the registration is successful then communicate the returned account with the activity and replace the current fragment with the Account Fragment.
    c. If the registration is not successful, show a Toast message indicating that the login was not successful.
    d. If there is missing input, show a Toast indicating missing input.
2. Clicking “Cancel” should replace this fragment with the Login Fragment.

Part 3 : Account Fragment (30 Points)

This greets the logged in user as shown in Figure 2(c), The requirements are as follows:

1. This fragment should receive the currently logged in user’s account from the Activity, and should use this account to display the user’s name in the greeting as shown in Figure 2(c).
2. Clicking the “Edit Profile” button should perform the following tasks:
    a. Replace the current fragment with the Update Account Fragment.
    b. Push the current fragment on the back stack.
    c. This should all be performed through the Main Activity.
3. Clicking the “Logout” button should delete the account stored in the Main Activity, and replace this fragment with the Login Fragment.

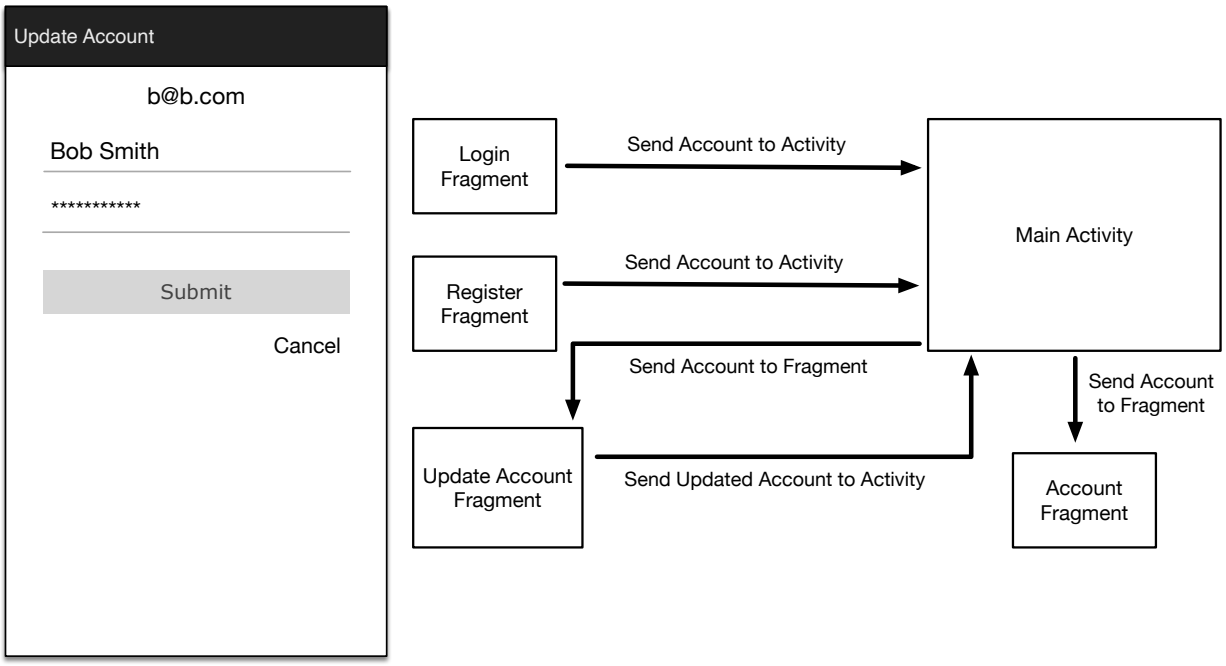
Part 4 : Update Account Fragment (30 Points)

The interface should be created to match Figure 3(a). The requirements are as follows:

1. This fragment should receive the currently logged in user’s account from the Activity, and should allow the user to update the user name, and password.
2. Clicking the “Cancel” button should dismiss this fragment and should pop the Account fragment from the back stack.
3. Clicking the “Submit” button, the app should check if all the fields are not empty and should display a Toast if there are any errors.

a. If all the fields are entered, the account should be updated using the DataServices.update function. If the update is successful, the newly updated account should be communicated with the Main Activity, should dismiss this fragment and should pop the Account fragment from the back stack. Upon returning to the Account fragment, the updated name should be displayed to reflect the account change.

4. Figure 3(b) shows the expected data flow between the different fragments and the Main Activity



(a) Update Account Fragment (b) Fragment and Activity Flow

Figure 3, App Wireframe and Flow