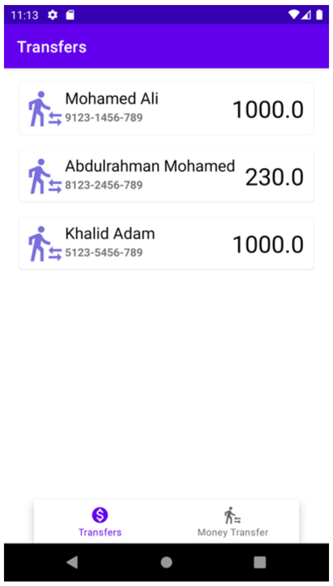
Lab8-Interacting with Web API using Coroutines and RetrofitCMPS 312
Mobile App Development
Lab 8 – Web API with Coroutines and Retrofit

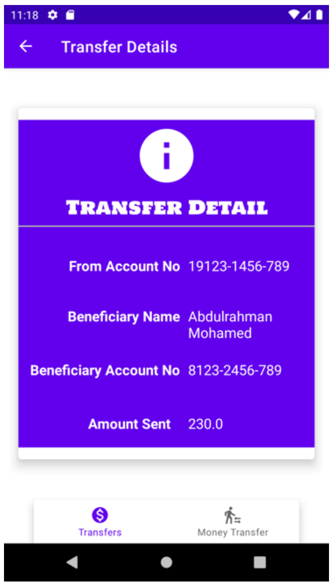
Objective

In this Lab, you will **continue building the Banking App** and make the app communicate with Web API. You will be using retrofit library in conjunction with coroutines to get, add, update, and delete transfers and beneficiaries.

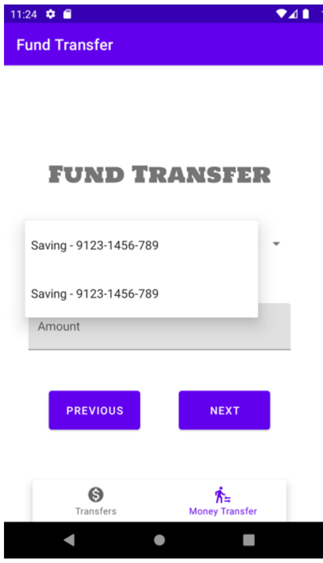
In this Lab you will practice interacting with Web API using asynchronous suspend functions and coroutines as well as Retrofit library.



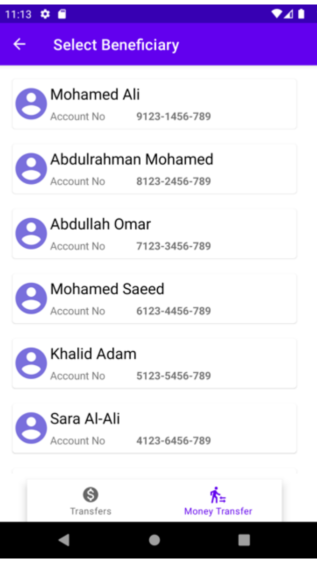
Mohamed Ali 9123-1456-789 1000.0
Abdulrahman Mohamed 8123-2456-789 230.0
Khalid Adam 5123-5456-789 1000.0



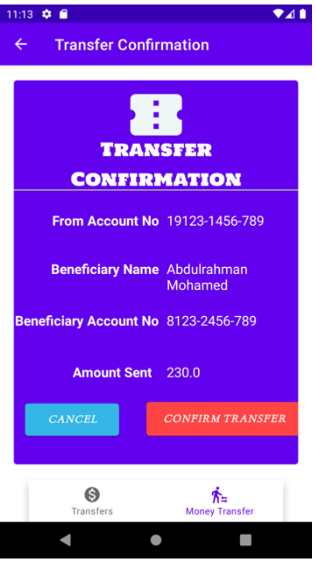
From Account No 19123-1456-789
Beneficiary Name Abdulrahman Mohamed
Beneficiary Account No 8123-2456-789
Amount Sent 230.0



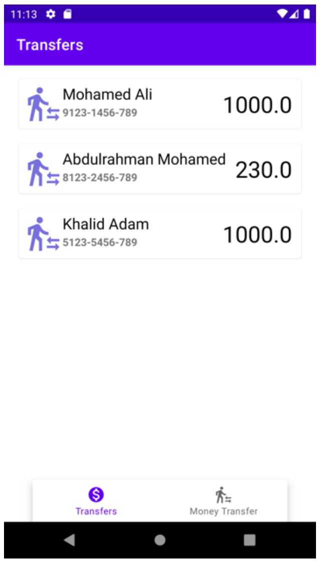
Saving - 9123-1456-789
Saving - 9123-1456-789
Amount



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



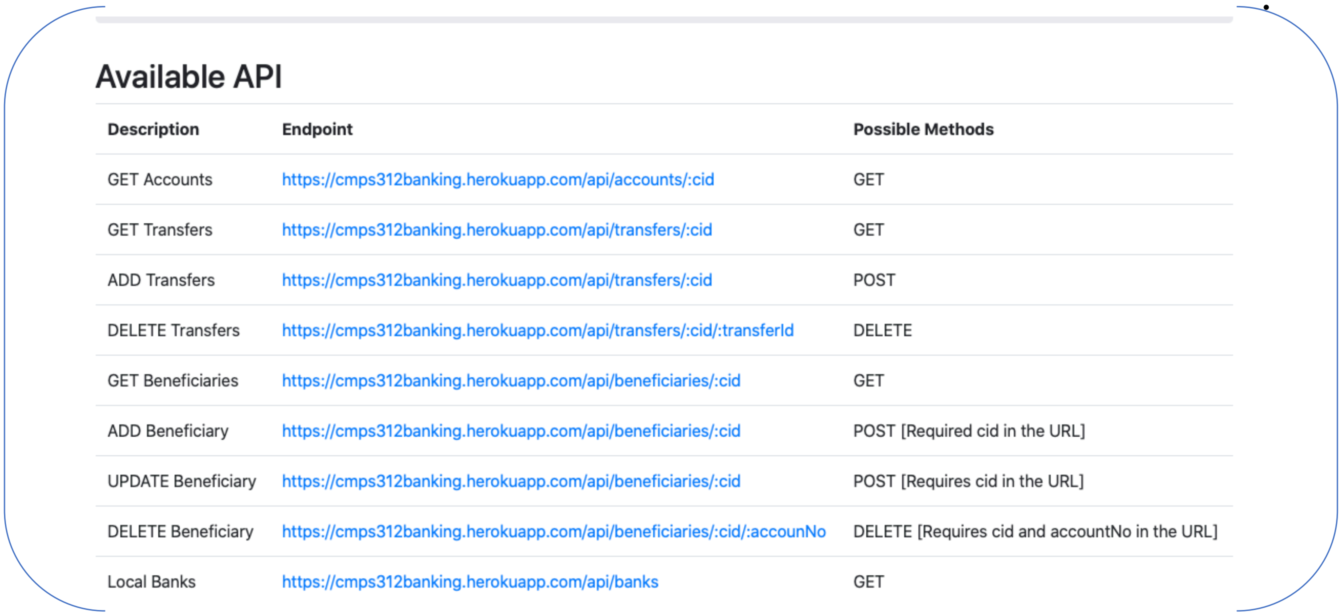
From Account No 19123-1456-789
Beneficiary Name Abdulrahman Mohamed
Beneficiary Account No 8123-2456-789
Amount Sent 230.0



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

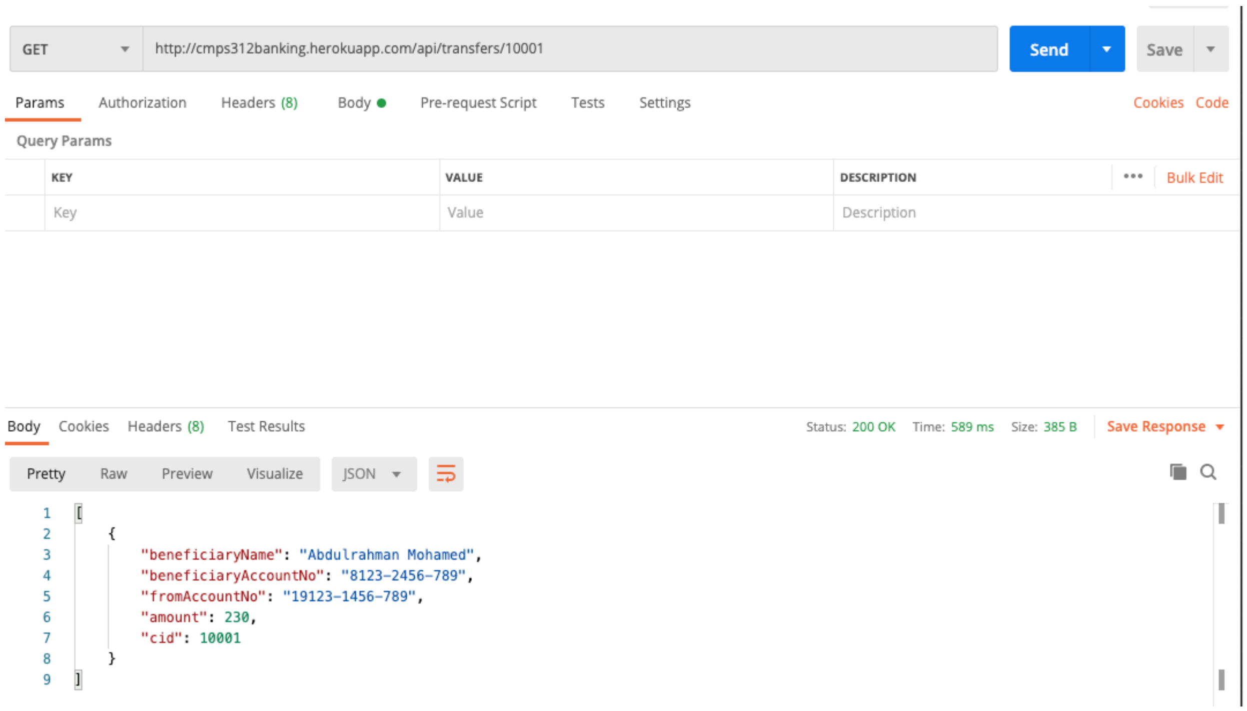
Preparation

1. Sync the Lab GitHub repo and copy the **Lab 8-Web API with Coroutines** folder into your repository.
2. Download postman from https://www.postman.com/downloads/ and test the following Banking Service Web API available at https://cmps312banking.herokuapp.com



Available API

Description	Endpoint	Possible Methods
GET Accounts	https://cmps312banking.herokuapp.com/api/accounts/:cid	GET
GET Transfers	https://cmps312banking.herokuapp.com/api/transfers/:cid	GET
ADD Transfers	https://cmps312banking.herokuapp.com/api/transfers/:cid	POST
DELETE Transfers	https://cmps312banking.herokuapp.com/api/transfers/:cid/:transferId	DELETE
GET Beneficiaries	https://cmps312banking.herokuapp.com/api/beneficiaries/:cid	GET
ADD Beneficiary	https://cmps312banking.herokuapp.com/api/beneficiaries/:cid	POST [Required cid in the URL]
UPDATE Beneficiary	https://cmps312banking.herokuapp.com/api/beneficiaries/:cid	POST [Requires cid in the URL]
DELETE Beneficiary	https://cmps312banking.herokuapp.com/api/beneficiaries/:cid/:accounNo	DELETE [Requires cid and accountNo in the URL]
Local Banks	https://cmps312banking.herokuapp.com/api/banks	GET



http://cmps312banking.herokuapp.com/api/transfers/10001

KEY	VALUE	DESCRIPTION
Key	Value	Description

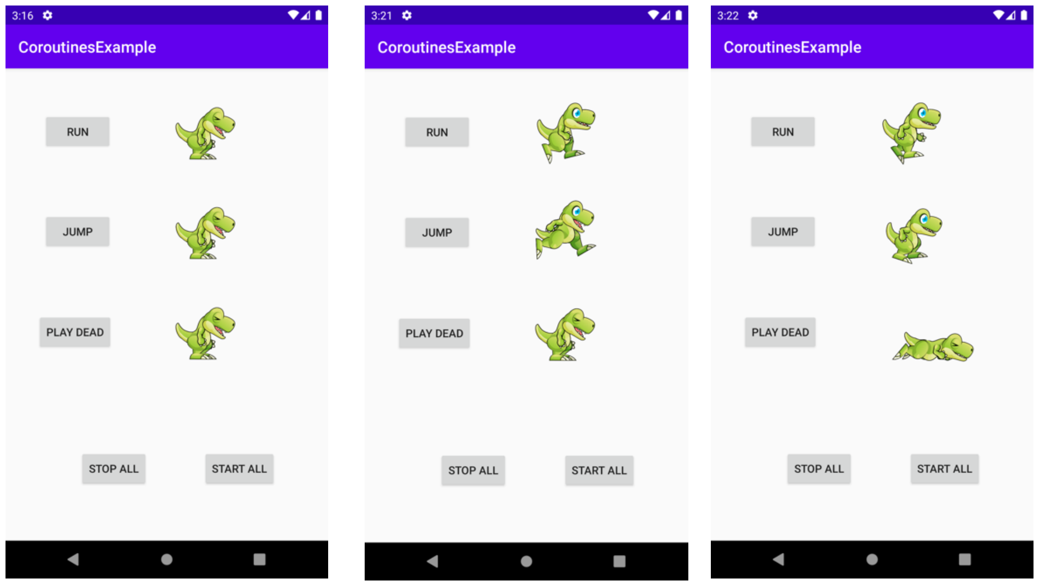
Status: 200 OK Time: 589 ms Size: 385 B

```
[
    {
        "beneficiaryName": "Abdulrahman Mohamed",
        "beneficiaryAccountNo": "8123-2456-789",
        "fromAccountNo": "19123-1456-789",
        "amount": 230,
        "cid": 10001
    }
]
```

PART A: Coroutines warm-up app

Open **Coroutines*ExampleApp*** project Android Studio. The ***Banking App*** project has the complete implementation of **Lab7-BankingApp** with very minor modifications such as swipe to delete and new properties added to the Account class such as cid (i.e., Customer id).

This lab activity will help you understand how coroutines work.



1. Write three functions named **run**, **jump** and **playDead** that load the sprits from the drawable folder. You can use the following code to load the images.

```
val imgeId = resources.getIdentifier(imageName, "drawable", packageName)
jumpIv.setImageResource(imgeId)
```

2. Then run a loop inside the three functions and animate the images. Below is the complete code that you need inside the jump function. Do the same for the other two functions [run and playDead]. Also, do not forget to add the log message.

```
repeat(1000) {
    val imgeId = resources.getIdentifier("jump${it % 12 + 1}", "drawable", packageName)
    jumpIv.setImageResource(imgeId)
    Log.d(TAG, "jump: ")
}
```

3. Call the three functions when the Start All button is clicked and see if the app will be able to animate the images.

   Question : How smooth is the animation ?

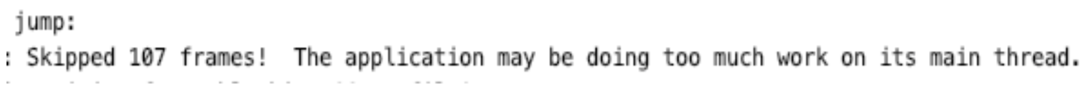
4. Let us try to improve the smoothness of the animation by using coroutines. The first coroutine scope we will see is the **GlobalScope**. So write the following code inside the onCreateMethod

```
GlobalScope.launch {
      delay(1000) //suspend without blocking
      //delay(5000) //suspend without blocking
      Log.d(TAG, "Coroutine Thread Name ${Thread.currentThread().name}")
}
Log.d(TAG, "Main Thread Name ${Thread.currentThread().name}")
```

Question : What do you understand from the names of the thread?

5. Try to execute your methods inside the coroutine. **How is the animation now?** Better but still not smooth right?
6. Now let us make it even better by making the three methods **suspend** method.

```
suspend fun run()
```

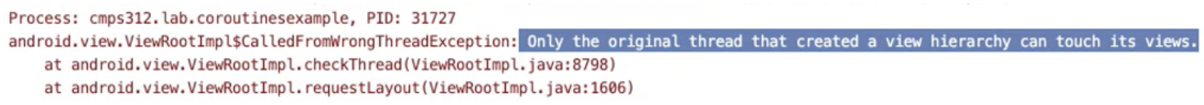


```
jump:
: Skipped 107 frames!  The application may be doing too much work on its main thread.
```

7. Make your suspend functions [**jump** , **run** and **playDead**] run in the IO context.

```
suspend fun run() = withContext(Dispatchers.IO)
suspend fun playDead() = withContext(Dispatchers.IO)
```

When you run your app you should see this error because you are trying to modify the UI elements from a another thread which the coroutine is running at.



```
Process: cmps312.lab.coroutinesexample, PID: 31727
android.view.ViewRootImpl$CalledFromWrongThreadException: Only the original thread that created a view hierarchy can touch its views.
    at android.view.ViewRootImpl.checkThread(ViewRootImpl.java:8798)
    at android.view.ViewRootImpl.requestLayout(ViewRootImpl.java:1606)
```

8. You can fix this by witching the context when assigning the image. See the below code

```
withContext(Dispatchers.Main) {
    runImg.setImageResource(imgeId)
}
```

9. Try to make the three functions run in parallel, by launching three co-routiens
10. Try to stop all the coroutines when the user presses the stop all button
11. Compare the lifecycleScope, GlobalScope by adding this code to the pause

```
override fun onPause() {
    finish()
    super.onPause()
}
```

Question : What do you see in the log when you finish the main activity while using the GlobalScope vs lifeCycleScope?

PART B: Implementing the Service APIs

Open the ***Banking App*** project on Android Studio. This project has the complete implementation of **Lab7-BankingApp** with some minor modifications such as swipe to delete and new properties added to the Account class such as cid (i.e., Customer id).

Your task is to interact with the remote Banking Service Web API to read/write data from/to the remote service. You will be using retrofit with coroutines to achieve this.

1. Add the following necessary dependencies for retrofit and coroutines in your graidle app module.

```
//retrofit library
implementation 'com.squareup.retrofit2:retrofit:2.9.0'
//this is to enable as to use the Kotlin Serlization
implementation("com.jakewharton.retrofit:retrofit2-kotlinx-serialization-
converter:0.7.0")
// toMediaType() when using app/json
implementation 'com.squareup.okhttp3:okhttp:4.9.0'

def lifecycle_version = "2.2.0"
// ViewModel
implementation "androidx.lifecycle:lifecycle-viewmodel-ktx:$lifecycle_version"
// LiveData
implementation "androidx.lifecycle:lifecycle-livedata-ktx:$lifecycle_version"
```

2. Add the internet permission inside your **AndroidManifest.xml** file or your app will not be allowed to access the network.

   ```
   <uses-permission android:name="android.permission.INTERNET"/>
   ```

3. Inside the **model** package create two sub-packages called **api** and **repository.**

4. Inside the **api** package create an `interface` called **BankService**

   Add all the interfaces methods that allows the app to communicate with Banking Service API available at https://cmps312banking.herokuapp.com

   Example : the following `getAccounts()` method allows sending a get request to Url https://cmps312banking.herokuapp.com/api/accounts/100101 and return the list of accounts for customer 10001.

   ```
   @GET("accounts/{cid}")
   suspend fun getAccounts(@Path("cid") cid : Int) : List<Account>
   ```

   Now implement the remaining **eight** methods.

   ```
   [getTransfers , addTransfer , deleteTransfer , getBeneficiaries
   addBeneficiary , updateBeneficiary , deleteBeneficiary]
   ```

5. Create an Kotlin file under **model/repository** package and name it **BankRepository.**

   In the **BankRepository** object declare the following three properties

   ```
   // The customerId is hardcoded for simplification. However, in a real app it will
   extracted from the user profile after login.

   val customerID = 10001
   ```

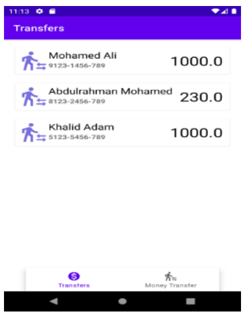
```
private const val BASE_URL = "https://cmps312banking.herokuapp.com/api/"
private val contentType = "app/json".toMediaType()
val jsonConverterFactory = Json { ignoreUnknownKeys = true
                                    }.asConverterFactory(contentType)

//this will instantiate the retrofit instance that will allow the app to perform the crud
operation
 val bankService by lazy {
    Retrofit.Builder()
        .baseUrl(BASE_URL)
        .addConverterFactory(jsonConverterFactory)  //json to kotlin object and vice versa
        .build()
        .create(BankService::class.java)
}
```

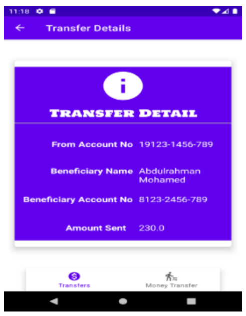
PART C: Linking the App View Models with the Repository

In PART C your task is to replace the **old repository** with the **new repository** that uses the retrofit library.

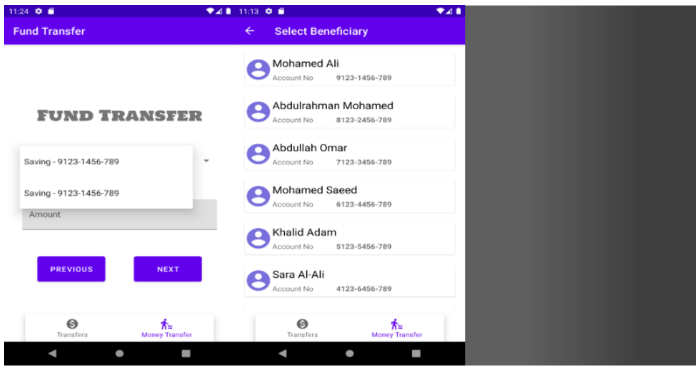
1. Modify the **TransferViewModel's** accounts, **_transfers** to read the transfer from the Banking Service API. You should use the **retrofit instance** that you created inside the data/repository/BankRepository object.
2. Do the same for the **BeneficiaryViewModel** and fetch the beneficiaries from the Web API instead of the assets folder. You should not change anything else in the app and it should work as before. This is the fruit of MVVM!!! 👍👌



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

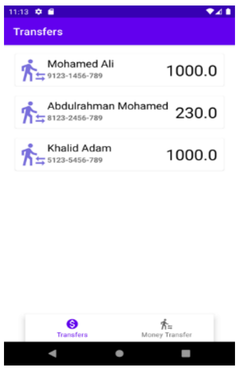


From Account No 19123-1456-789
Beneficiary Name Abdulrahman Mohamed
Beneficiary Account No 8123-2456-789
Amount Sent 230.0



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

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



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