[Android] How-to: Integrate a Flow Journey into the Productized App

Introduction

Backbase Digital Sales Onboarding offers a series of Flow journeys and other common Flow components that, put together, allow a user to be onboarded with a new account. In this exercise, we will demonstrate how to incorporate these journeys into the existing productized Retail App so that users will be able to either log in to an existing account (as is in the existing version of the productized app), as well as sign up if they do not have an account.

Pre-requisites

Before setting up your `gradle` files, remember that some of the Flow Journeys dependencies are intended as custom journeys and should be imported from the "US onboarding reference app" as external modules. If you need access to the reference repo, reach out to the Digital Sales team.

In addition, since the Flow app is not productized, you will need to include the necessary app configurations and routers from the reference app into your project.

Your `settings.gradle` file will need to include references to each of the necessary dependencies

settings.gradle

```
include ':app'

include ':walkthrough'
include ':onboarding'
include ':credentials'
include ':kyc'
include ':declined'
include ':aboutyou'
include ':ssn'
include ':in-review'
include ':done'
include ':coapplicant-welcome'
include ':coapplicant'
include ':loading'
include ':identity-verified'
include ':welcome'

```

Then, your project `build.gradle` file needs to have all the flow dependencies added.

build.gradle (Project)

```
flow                      : "com.backbase.android.flow:flow-interaction-sdk:3.0.0”,
flowotp                   : "com.backbase.android.flow.otp:one-time-password-journey:3.1.0",
flowaddressvalidation     : "com.backbase.android.flow.address:address-validation-journey:3.0.0",
flowidentityverification: "com.backbase.android.flow.identityverification:identity-verification-journey:3.0.
flowbeneficiaries         : "com.backbase.android.flow.beneficiary:beneficiaries-journey:0.1.0-beta01",
flowproductselection      : "com.backbase.android.flow.productselector:product-selection-journey:0.2.0",
flowstepnavigation        : "com.backbase.android.flow.stepnavigation:step-navigation-journey:0.1.1-beta01",
flowcommon                : 'com.backbase.android.flow.common:flow-common:1.1.7',
flowdesign                : 'com.backbase.android.flow.design:flow-design:1.0.0',
flowformbuilder           : 'com.backbase.android.flow.formbuilder:flow-form-builder:1.2.0',
```

And finally, in the `build.gradle` app module, add implementations for the US onboarding external modules.

build.gradle (app module)

```
implementation project(path: ':userprofile')
implementation project(path: ':walkthrough')
implementation project(path: ':aboutyou')
implementation project(path: ':ssn')
implementation project(path: ':onboarding')
implementation project(path: ':credentials')
implementation project(path: ':kyc')
implementation project(path: ':identity-verified')
implementation project(path: ':declined')
implementation project(path: ':in-review')
implementation project(path: ':done')
implementation project(path: ':coapplicant-welcome')
implementation project(path: ':coapplicant')
implementation project(path: ':loading')
implementation project(path: ':welcome')
```

Setting up the Landing Journey

To demonstrate how we can provide users with two separate user actions, we will be implementing a custom journey called Landing. This journey will consist of a simple screen that contains two buttons, a ‘Sign Up’ and a ‘Log In’ button.

If the user already has an existing account, the `Log In` button will lead them to the normal Retail App authentication flow through Identity. Otherwise, the `Sign Up` button will route them to register for an account using the Digital Sales Onboarding Flow.

Creating the Landing layout

When creating your own custom view, you should take advantage of the Backbase Mobile Design System to ensure a consistent look and feel across the app. Here, we will create a custom XML file to be used for `Sign Up` and `Log In` buttons.

fragment_landing.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#006CBF">

  <com.backbase.android.design.button.BackbaseButton
      android:id="@+id/btn_signup"
      style="?attr/buttonStyleSecondary"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginBottom="25dp"
      android:backgroundTint="@color/white"
      android:text="@string/sign_up"

      app:layout_constraintBottom_toTopOf="@id/btn_login"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintStart_toStartOf="parent" />

  <com.backbase.android.design.button.BackbaseButton
      android:id="@+id/btn_login"
      style="?attr/buttonStyleSecondary"
```

```
        android:text="@string/login"

        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="25dp"
        android:backgroundTint="@color/white"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

We will also add the buttons texts to our `strings.xml` file

strings.xml

```
...

<string name="sign_up">Sign Up</string>
<string name="login">Log In</string>

...
```

Inserting the landing screen in our app.

To navigate to our newly added landing screen, we have to change the current navigation graph after the Splash screen animation is finished to avoid going directly towards the login screen by changing the route to our custom landing screen.

navigation/us_app.xml

```
...
<fragment
    android:id="@+id/splashScreen"
    android:name="com.backbase.android.retail.journey.app.common.splash.SplashScreen"
    android:label="Splash screen"

    tools:layout="@layout/splash_screen">
  <action
      android:id="@+id/action_splashScreen_to_app"
      app:destination="@id/landingFragment" />
</fragment>

...
```

Also, we have to add the landing fragment to the navigation graph and create the routes to the `Onboarding Flow` and to the `Login Screen`

navigation/us_app.xml

```
...

<fragment
    android:id="@+id/landingFragment"
   android:name="com.backbase.cse.retailapp.landing.LandingFragment"
    android:label="LandingFragment" >
  <action
      android:id="@+id/action_landingFragment_to_authenticationJourney"
      app:destination="@id/authenticationJourney" />
  <action
      android:id="@+id/action_landingFragment_to_onboardingScreen"
      app:destination="@id/onboardingScreen" />
</fragment>

...
```

Using the above layout, we can now create our View for the Landing Journey. On each of the layout buttons, we are adding an action to navigate towards our `Sign Up`, respectively `Log In` destinations.

LandingFragment.kt

```
import android.os.Bundle
import android.view.View
import androidx.fragment.app.Fragment
import androidx.navigation.fragment.findNavController
import com.backbase.cse.R
import kotlinx.android.synthetic.main.fragment_landing.btn_login
import kotlinx.android.synthetic.main.fragment_landing.btn_signup

class LandingFragment : Fragment(R.layout.fragment_landing) {

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        btn_login.setOnClickListener {
            findNavController().navigate(R.id.action_landingFragment_to_authenticationJourney)
        }

        btn_signup.setOnClickListener {
            findNavController().navigate(R.id.action_landingFragment_to_onboardingScreen)
        }

    }
}
```

Navigating out of the Landing Journey

Preparing to enter Digital Sales Flow

Let’s break down what we want to accomplish in the Sign Up flow. For the Digital Sales Onboarding flow to be successful, you need to set up the following items:

- Accessing the values in the correct configuration file
- Registering your Flow dependencies (journeys)
- Creating the onboarding journey router.

Setting up the Flow configuration file

Because Digital Sales may run in a separate environment, you will want to have a separate configuration file to contain the references to the correct server URLs and other miscellaneous values that are necessary to start the onboarding process.

config.json

```
{
  "development": {
    "debugEnable": false
  },
  "backbase": {
    "serverURL": "<your-server-url>",
    "localModelPath": "$(contextRoot)/localModel.json",
    "experience": "",
    "version": "6.1.5",
    "identity": {
      "baseURL": <your-identity-server-url>,
      "realm": "retail",
      "clientId": "mobile-client",
      "applicationKey": "retail"
    }
  },
  },
  "security": {
    "sslPinning": {
      "checkChain": true
    }
  }
}

```

We will also need to create a configuration that pulls the data correctly from this configuration file. As such, you will need to initialize Backbase with the config asset path.

MyApplication.kt

```kotlin
private fun initBackbase() {
    initializeBackbase(
        backbaseConfigAssetPath = "backbase/config.json", false
    ).also {
        setHttpHeaders()

    }
}
```

Registering your Flow dependencies (journeys)

We will have to register our Flow Journey configurations and add all the external modules to define all the app-level Authentication Journey configurations.

ApplicationModule.kt

```kotlin
import android.content.Context
import com.backbase.android.Backbase
import com.backbase.android.dbs.dataproviders.AssetsFileDBSDataProvider
import com.backbase.android.dbs.dataproviders.NetworkDBSDataProvider
import com.backbase.android.flow.aboutyou.AboutYouConfiguration
import com.backbase.android.flow.aboutyou.aboutYouJourneyModule
import com.backbase.android.flow.aboutyou.usecase.AboutYouUseCase
import com.backbase.android.flow.aboutyou.usecase.AboutYouUseCaseDefaultImpl
import com.backbase.android.flow.address.AddressConfiguration
import com.backbase.android.flow.address.addressJourneyModule
import com.backbase.android.flow.address.usecase.AddressUseCase
import com.backbase.android.flow.address.usecase.AddressUseCaseDefaultImpl
import com.backbase.android.flow.coapplicant.CoApplicantConfiguration
import com.backbase.android.flow.coapplicant.coApplicantJourneyModule
import com.backbase.android.flow.coapplicant.usecase.CoApplicantUseCase
import com.backbase.android.flow.coapplicant.usecase.CoApplicantUseCaseDefaultImpl
import com.backbase.android.flow.coapplicant.welcome.CoApplicantWelcomeConfiguration
import com.backbase.android.flow.coapplicant.welcome.coApplicantWelcomeJourneyModule
import com.backbase.android.flow.coapplicant.welcome.usecase.CoApplicantWelcomeUseCase
import com.backbase.android.flow.coapplicant.welcome.usecase.CoApplicantWelcomeUseCaseDefaultImpl
import com.backbase.android.flow.common.interaction.OnboardingBaseUseCase
import com.backbase.android.flow.common.interaction.OnboardingBaseUseCaseDefaultImpl
import com.backbase.android.flow.common.v2.interaction.OnboardingBaseUseCase as OnboardingBaseUseCase2
import com.backbase.android.flow.common.v2.interaction.OnboardingBaseUseCaseDefaultImpl as OnboardingBaseUs
import com.backbase.android.flow.contracts.FlowClientContract
import com.backbase.android.flow.v2.contracts.FlowClientContract as FlowClientContract2
import com.backbase.android.flow.v2.FlowClient as FlowClient2
import com.backbase.android.flow.FlowClient
import com.backbase.android.flow.common.model.OnboardingModel
import com.backbase.android.flow.common.utils.readAsset
import com.backbase.android.flow.credentials.CredentialsConfiguration
import com.backbase.android.flow.credentials.CredentialsJourneyModule
import com.backbase.android.flow.credentials.usecase.CredentialsUseCase
import com.backbase.android.flow.credentials.usecase.CredentialsUseCaseDefaultImpl
import com.backbase.android.flow.declined.DeclinedConfiguration
import com.backbase.android.flow.declined.declinedJourneyModule
import com.backbase.android.flow.declined.models.DeclinedModel
import com.backbase.android.flow.declined.usecase.DeclinedUseCase
import com.backbase.android.flow.declined.usecase.DeclinedUsecaseDefaultImpl
import com.backbase.android.flow.done.doneJourneyModule
import com.backbase.android.flow.done.usecase.DoneUseCase
import com.backbase.android.flow.done.usecase.DoneUseCaseDefaultImpl
import com.backbase.android.flow.identityverification.DocScannerDataCenter
import com.backbase.android.flow.identityverification.IdentityVerificationConfiguration
import com.backbase.android.flow.identityverification.IdentityVerificationJourneyModule
import com.backbase.android.flow.identityverification.usecase.IdentityVerificationUseCase
import com.backbase.android.flow.identityverification.usecase.IdentityVerificationUseCaseDefaultImpl
import com.backbase.android.flow.identityverified.IdentityVerifiedConfiguration
import com.backbase.android.flow.identityverified.identityVerifiedJourneyModule
import com.backbase.android.flow.identityverified.usecase.IdentityVerifiedUseCase
import com.backbase.android.flow.identityverified.usecase.IdentityVerifiedUseCaseDefaultImpl
import com.backbase.android.flow.inreview.inReviewJourneyModule
import com.backbase.android.flow.inreview.usecase.InReviewUseCase
import com.backbase.android.flow.inreview.usecase.InReviewUsecaseDefaultImpl
import com.backbase.android.flow.kyc.KycConfiguration
import com.backbase.android.flow.kyc.kycJourneyModule
import com.backbase.android.flow.kyc.usecase.KycUsecase
import com.backbase.android.flow.kyc.usecase.KycUsecaseDefaultImpl
import com.backbase.android.flow.loading.*
import com.backbase.android.flow.loading.usecase.LoadingUseCase
import com.backbase.android.flow.loading.usecase.LoadingUseCaseDefaultImpl
import com.backbase.android.flow.onboarding.OnboardingConfiguration
import com.backbase.cse.retailapp.OnboardingConstants.Companion.DBS_PATH
import com.backbase.cse.retailapp.OnboardingConstants.Companion.JUMIO_API_SECRET
import com.backbase.cse.retailapp.OnboardingConstants.Companion.JUMIO_API_TOKEN
import com.backbase.cse.retailapp.OnboardingConstants.Companion.ONBOARDING_INTERACTION_NAME
import com.backbase.cse.retailapp.OnboardingConstants.Companion.SERVICE_NAME
import com.backbase.cse.retailapp.OnboardingConstants.Companion.SHARED_CONTENT_REPOSITORY
import com.backbase.cse.retailapp.usecase.WelcomeJourneyUseCaseImpl
import com.backbase.cse.retailapp.walkthrough.securePage
import com.backbase.cse.retailapp.walkthrough.termsAndConditionsPages
import com.backbase.cse.retailapp.walkthrough.welcomePage
import com.backbase.android.flow.onboarding.models.Step
```

```kotlin
import com.backbase.android.flow.onboarding.onboardingModule
import com.backbase.android.flow.otp.OtpConfiguration
import com.backbase.android.flow.otp.otpJourneyModule
import com.backbase.android.flow.otp.usecase.OtpUseCase
import com.backbase.android.flow.otp.usecase.OtpUseCaseDefaultImpl
import com.backbase.android.flow.productselector.*
import com.backbase.android.flow.walkthrough.usecase.WalkthroughUseCase
import com.backbase.android.flow.walkthrough.usecase.WalkthroughUseCaseDefaultImpl
import com.backbase.android.flow.walkthrough.walkthroughJourneyModule
import com.backbase.android.flow.ssn.SsnConfiguration
import com.backbase.android.flow.ssn.ssnJourneyModule
import com.backbase.android.flow.ssn.usecase.SsnUseCase
import com.backbase.android.flow.ssn.usecase.SsnUseCaseDefaultImpl
import com.backbase.android.flow.walkthrough.WalkthroughConfiguration
import com.backbase.android.flow.welcome.WelcomeJourneyUseCase
import com.backbase.android.flow.welcome.welcomeJourneyModule
import com.backbase.android.flow.welcome.welcomeScreenConfiguration
import com.backbase.cse.R
import com.backbase.deferredresources.DeferredText
import com.google.gson.Gson
import com.google.gson.reflect.TypeToken
import org.koin.core.context.loadKoinModules
import org.koin.dsl.module
import java.lang.reflect.Type
import java.net.URI

val applicationModule = module {

    single {
        Gson()
    }

    single {
        NetworkDBSDataProvider(get())
    }

    factory {
        AssetsFileDBSDataProvider(get())
    }

    val serverURL = Backbase.getInstance()?.configuration?.experienceConfiguration?.serverURL
    val baseURI = "$serverURL/$DBS_PATH/$SERVICE_NAME"
    single<FlowClientContract>() {
        FlowClient(
            get(),
            URI(baseURI),
            get<NetworkDBSDataProvider>(),
            null,
            ONBOARDING_INTERACTION_NAME,
            null
        )
    }
    single<FlowClientContract2>() {
        FlowClient2(
            context = get(),
            baseUri = URI(baseURI),
            dataProvider = get<NetworkDBSDataProvider>(),
            interactionName = ONBOARDING_INTERACTION_NAME,
            interactionResponseDefaultBodyType = OnboardingModel::class.java
        )
    }

    factory<OnboardingBaseUseCase2> {
        OnboardingBaseUseCaseDefaultImpl2(get())
    }

    factory<OnboardingBaseUseCase> {
        OnboardingBaseUseCaseDefaultImpl(get())
    }
    //endregion

    // region Beneficiary Journey
    factory<ProductSelectorUseCase>{
        ProductSelectorUseCaseDefaultImpl(get(), get())
    }

    factory {
        ProductSelectorConfiguration {
            imageBaseUrl = serverURL.toString()
            requestProductsAction = "get-product-list"
            submitProductAction = "select-products"
            createCaseAction = ""
            selectionType = SelectionType.SINGLE
        }
    }

    loadKoinModules(listOf(ProductSelectorJourneyModule))

    //endregion

    // region Co-Applicant Journey
    factory<CoApplicantUseCase> {
        CoApplicantUseCaseDefaultImpl(get(), get())
    }

    factory {
        val context: Context by inject()
        CoApplicantConfiguration {
            coApplicantActionName = "account-type-selector"
        }
    }

    loadKoinModules(listOf(coApplicantJourneyModule))

    //endregion

    // region Co-Applicant Welcome Journey
    factory<CoApplicantWelcomeUseCase> {
        CoApplicantWelcomeUseCaseDefaultImpl(get(), get())
    }

    factory {
        val context: Context by inject()
        CoApplicantWelcomeConfiguration {
            coApplicantFetchDataActionName = "fetch-co-applicant-data"
            coApplicantAttachActionName = "fetch-co-applicant"
```

```kotlin
            coApplicantWelcomeInteractionName = "co-applicant-welcome"
            onboardingInteractionName = ONBOARDING_INTERACTION_NAME
        }
    }

    loadKoinModules(listOf(coApplicantWelcomeJourneyModule))

    //endregion

    //region Loading Journey
    factory<LoadingUseCase> {
        LoadingUseCaseDefaultImpl(get(), get())
    }

    factory {
        LoadingConfiguration {
            initializeActionName = "fetch-co-applicant"
        }
    }
    loadKoinModules(listOf(loadingJourneyModule))
    //endregion

    // region Welcome Journey
    factory<WelcomeJourneyUseCase> {
        WelcomeJourneyUseCaseImpl()
    }

    factory {
        val context: Context by inject()
        welcomeScreenConfiguration {
            brandTitle = DeferredText.Constant(context.getString(R.string.brand_title))
            brandTitleDetail = DeferredText.Constant(context.getString(R.string.brand_title_detail))
        }
    }

    loadKoinModules(listOf(welcomeJourneyModule))

    //endregion

    //region Walkthrough journey

    factory {
        WalkthroughConfiguration {
            pages = arrayListOf(
                welcomePage(),
                securePage(get()),
                termsAndConditionsPages()
            )
            privacyPolicyUrl = "https://www.backbase.com/privacy-policy/"
            termsAndConditionsUrl = "https://www.backbase.com/legal/"
            submitTermsAndConditionsActionName = "agree-to-terms"
            agreedToTermsPayloadKey = "agreedToTerms"
        }
    }

    factory<WalkthroughUseCase> {
        WalkthroughUseCaseDefaultImpl(get(), get())
    }

    loadKoinModules(listOf(walkthroughJourneyModule))

    //endregion

    //region onboarding journey
    factory {
        AboutYouConfiguration {
            val gson: Gson by inject()
            val context: Context by inject()
            val formItemsType: Type = object :
                TypeToken<ArrayList<com.backbase.android.flow.aboutyou.models.FormItem>>() {}.type
            formItems = gson.fromJson(
                readAsset(
                    context.assets,
                    "backbase/onboarding/anchor.json"
                ), formItemsType
            )
            submitActionName = "submit-anchor-data"
            prefillActionName = "prefill-anchor-data"
        }
    }

    factory<AboutYouUseCase> {
        AboutYouUseCaseDefaultImpl(get(), get())
    }

    loadKoinModules(listOf(aboutYouJourneyModule))
    //endregion

    //region ssn journey
    factory {
        SsnConfiguration {
            val gson: Gson by inject()
            val context: Context by inject()
            val formItemsType: Type = object :
                TypeToken<ArrayList<com.backbase.android.flow.ssn.models.FormItem>>() {}.type
            formItems = gson.fromJson(
                readAsset(
                    context.assets,
                    "backbase/onboarding/ssn.json"
                ), formItemsType
            )
            submitSsnActionName = "submit-ssn"
            selectCitizenshipActionName = "citizenship-selector"
        }
    }

    factory<SsnUseCase> {
        SsnUseCaseDefaultImpl(get(), get())
    }

    loadKoinModules(listOf(ssnJourneyModule))
    //endregion

    //region onboarding journey

    factory {
        OnboardingConfiguration {
            this.steps = steps
        }
    }

    loadKoinModules(listOf(onboardingModule))

    //endregion

    //region Address Validation Journey
    factory {
        AddressConfiguration {
            submitActionName = "submit-address"
            fetchActionName = "fetch-address"
        }
    }

    factory<AddressUseCase> {
        AddressUseCaseDefaultImpl(get(), get())
    }

    loadKoinModules(listOf(addressJourneyModule))
    //endregion

    //region OTP

    factory {
        OtpConfiguration {
            requestActionName = "request-otp"
            verifyActionName = "verify-otp"
            availableOtpChannelsActionName = "available-otp-channels"
            verificationCodeMaxLength = Integer(6)
        }
    }

    factory<OtpUseCase> {
        OtpUseCaseDefaultImpl(get(), get())
    }

    loadKoinModules(listOf(otpJourneyModule))
    //endregion

    //region
    loadKoinModules(listOf(CredentialsJourneyModule))

    factory {
        CredentialsConfiguration {
            actionName = "setup-credentials"
        }
    }

    factory<CredentialsUseCase> {
        CredentialsUseCaseDefaultImpl(get(), get())
    }
    //endregion

    //region identity_verified
    factory {
        IdentityVerifiedConfiguration {
            actionName = "empty-handler"
        }
    }

    factory<IdentityVerifiedUseCase> {
        IdentityVerifiedUseCaseDefaultImpl(get(), get())
    }

    loadKoinModules(listOf(identityVerifiedJourneyModule))
    //endregion

    //region successfully-done
    factory<DoneUseCase> {
        DoneUseCaseDefaultImpl(get())
    }
    loadKoinModules(listOf(doneJourneyModule))
    //endregion

    //region in-review-done
    factory<InReviewUseCase> {
        InReviewUsecaseDefaultImpl(get())
    }
    loadKoinModules(listOf(inReviewJourneyModule))
    //endregion

    //region declined

    factory {
        DeclinedConfiguration {
            declinedFileName = "/declinedScreen.json"
            contentRepoName = SHARED_CONTENT_REPOSITORY
            declinedDataModel = DeclinedModel(
                "http://www.backbase.com",
                "http://www.backbase.com",
                "http://www.backbase.com"
            )
        }
    }

    factory<DeclinedUseCase> {
        DeclinedUsecaseDefaultImpl(
            get(),
            get(),
            true
        )
    }

    loadKoinModules(listOf(declinedJourneyModule))

    //endregion

    //region kyc

    loadKoinModules(listOf(kycJourneyModule))

    factory {
        KycConfiguration {
            requestKycQuestionsActionName = "kyc-questions-loader"
            submitKycActionName = "submit-kyc-data"
        }
    }

    factory<KycUsecase> {
        KycUsecaseDefaultImpl(get(), get())
    }
    //endregion kyc

    //region IDV
    loadKoinModules(listOf(IdentityVerificationJourneyModule))

    factory {
        IdentityVerificationConfiguration {
            apiToken = JUMIO_API_TOKEN
            apiSecretKey = JUMIO_API_SECRET
            dataCenter = DocScannerDataCenter.EU
            initiationActionName = "identity-verification-initiation"
            verificationActionName = "identity-verification-result"
        }
    }

    factory<IdentityVerificationUseCase> {
        IdentityVerificationUseCaseDefaultImpl(get(), get())
    }
    //endregion IDV
}



```

And then we have to load the about application module in our `Application` file.

MyApplication.kt

```
...

override fun onCreate() {
        super.onCreate()
        initBackbase()

        koinApplication {
            loadKoinModules(applicationModule)
        }
}

...
```

Creating the onboarding journey router.

As we have imported all the journeys from our US onboarding app, we have to create a router for navigating between them in the case that our user wants to go back or to drop the onboarding process.

OnboardingRouter.kt

```
import android.content.Context
import androidx.navigation.NavController
```

```kotlin
import com.backbase.android.flow.FlowClient
import com.backbase.android.flow.common.model.ApplicantType
import com.backbase.android.flow.common.model.StepConfiguration
import com.backbase.android.flow.contracts.FlowClientContract
import com.backbase.android.flow.listeners.InteractionListener
import com.backbase.android.flow.onboarding.OnboardingRouter
import com.backbase.android.utils.net.response.Response
import com.backbase.cse.R
import kotlinx.coroutines.GlobalScope
import kotlinx.coroutines.launch
import kotlinx.coroutines.runBlocking
import org.koin.java.KoinJavaComponent.inject
import kotlin.coroutines.resume
import kotlin.coroutines.suspendCoroutine

fun onboardingRouter(
    navController: NavController,
    usecase: FlowClientContract,
    context: Context,
    completion: () -> Unit = {}
) = object : OnboardingRouter {
    override fun onClose() {
        resetInteractions(context)
        showJourneyWithClearStack(
            navController,
            R.id.onboardingScreen
        )
        completion()
    }

    private fun resetInteractions(context: Context) {
        val flowClient by inject(FlowClientContract::class.java)
        (flowClient as FlowClient)?.let {
            it.interactionId = null
            StepConfiguration.model = null
            StepConfiguration.interactionId = null
            StepConfiguration.stepName = null
            StepConfiguration.steps = null
            StepConfiguration.previousStepName = null
            ApplicantType.reset(context)
        }
    }

    fun isBackNavigationSupported(): Pair<Boolean, String?> {
        StepConfiguration.steps?.get(StepConfiguration.stepName)?.let { step ->
            step.back?.let { back ->
                return Pair(step.enabled, back)
            }
        }
        return Pair(false, null)
    }

    override fun goBack(): Boolean {
        val (isEnabled, back) = isBackNavigationSupported()
        if (isEnabled) {
            back?.let {
                StepConfiguration.stepName = back
                StepConfiguration.previousStepName = StepConfiguration.steps?.get(back)?.back
                return runBlocking {
                    suspendCoroutine<Boolean> {
                        GlobalScope.launch {
                            usecase.navigateToStep(back, object : InteractionListener<Object> {
                                override fun onSuccess(o: Object) {
                                    it.resume(true)
                                }

                                override fun onError(response: Response) {
                                    it.resume(response.responseCode == 200)
                                }
                            })
                        }
                    }
                }
            }
        }

        return false
    }
}

fun showJourneyWithClearStack(
    navController: NavController,
    journeyScreenResId: Int
) {
    val graph = navController.graph
    graph.startDestination = journeyScreenResId
    navController.graph = graph
}

```

After we have created the onboarding router, we have to add its dependency to our `Activity` class.

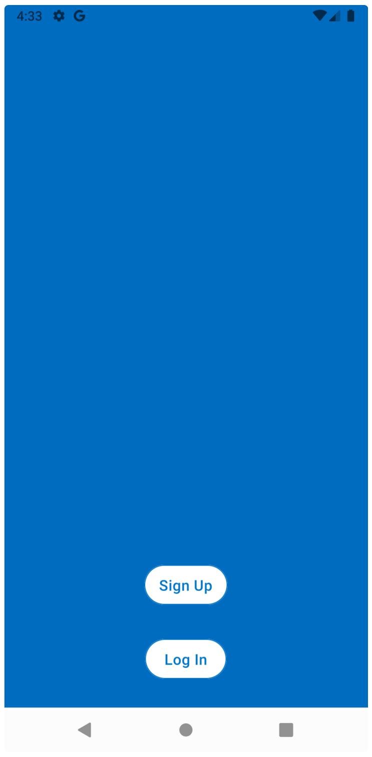
MyAppActivity.kt

```
…
private fun instantiateActivityModule() = module {
    factory {
        onboardingRouter(controller, get(), get()) {
            setTheme(R.style.AppTheme)
        }
    }
}
private val activityModule: Module by lazy { instantiateActivityModule() }

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    loadKoinModules(activityModule)
}
…
```

Conclusion

Following these implementation steps, you should now have a version of the Retail App that incorporates both the Digital Sales Flow and the Identity Authentication Journey as options for users launching the app. The Landing journey screen should look something like this, with buttons that route to their respective modules.


4:33
Sign Up
Log In