

[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

```
1 include ':app'
2
3 include ':walkthrough'
4 include ':onboarding'
5 include ':credentials'
6 include ':kyc'
7 include ':declined'
8 include ':aboutyou'
9 include ':ssn'
10 include ':in-review'
11 include ':done'
12 include ':coapplicant-welcome'
13 include ':coapplicant'
14 include ':loading'
15 include ':identity-verified'
16 include ':welcome'
17
```

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

▼ build.gradle (Project)

```
1 flow                : "com.backbase.android.flow:flow-interaction-sdk:3.0.0",
2 flowotp             : "com.backbase.android.flow.otp:one-time-password-journey:3.1.0",
3 flowaddressvalidation : "com.backbase.android.flow.address:address-validation-journey:3.0.0",
4 flowidentityverification : "com.backbase.android.flow.identityverification:identity-verification-journey:3.0.0",
5 flowbeneficiaries    : "com.backbase.android.flow.beneficiary:beneficiaries-journey:0.1.0-beta01",
6 flowproductselection : "com.backbase.android.flow.productselector:product-selection-journey:0.2.0",
7 flowstepnavigation   : "com.backbase.android.flow.stepnavigation:step-navigation-journey:0.1.1-beta01",
8 flowcommon           : "com.backbase.android.flow.common:flow-common:1.1.7",
9 flowdesign            : "com.backbase.android.flow.design:flow-design:1.0.0",
10 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)

```
1 implementation project(path: ':userprofile')
2 implementation project(path: ':walkthrough')
3 implementation project(path: ':aboutyou')
4 implementation project(path: ':ssn')
5 implementation project(path: ':onboarding')
6 implementation project(path: ':credentials')
7 implementation project(path: ':kyc')
8 implementation project(path: ':identity-verified')
9 implementation project(path: ':declined')
10 implementation project(path: ':in-review')
11 implementation project(path: ':done')
12 implementation project(path: ':coapplicant-welcome')
13 implementation project(path: ':coapplicant')
14 implementation project(path: ':loading')
15 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

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     android:background="#006CBF">
8
9     <com.backbase.android.design.button.BackbaseButton
10         android:id="@+id/btn_signup"
11         style="?attr/buttonStyleSecondary"
12         android:layout_width="wrap_content"
13         android:layout_height="wrap_content"
14         android:layout_marginBottom="25dp"
15         android:backgroundTint="@color/white"
16         android:text="@string/sign_up"
17
18         app:layout_constraintBottom_toTopOf="@id/btn_login"
19         app:layout_constraintEnd_toEndOf="parent"
20         app:layout_constraintStart_toStartOf="parent" />
21
22     <com.backbase.android.design.button.BackbaseButton
23         android:id="@+id/btn_login"
24         style="?attr/buttonStyleSecondary"
```

```

25     android:text="@string/login"
26
27     android:layout_width="wrap_content"
28     android:layout_height="wrap_content"
29     android:layout_marginBottom="25dp"
30     android:backgroundTint="@color/white"
31     app:layout_constraintBottom_toBottomOf="parent"
32     app:layout_constraintEnd_toEndOf="parent"
33     app:layout_constraintStart_toStartOf="parent" />
34
35 </androidx.constraintlayout.widget.ConstraintLayout>

```

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

▼ strings.xml

```

1  ...
2
3  <string name="sign_up">Sign Up</string>
4  <string name="login">Log In</string>
5
6  ...

```

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

```

1  ...
2  <fragment
3      android:id="@+id/splashScreen"
4      android:name="com.backbase.android.retail.journey.app.common.splash.SplashScreen"
5      android:label="Splash screen"
6
7      tools:layout="@layout/splash_screen">
8      <action
9          android:id="@+id/action_splashScreen_to_app"
10         app:destination="@id/landingFragment" />
11  </fragment>
12
13  ...

```

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

```

1  ...
2
3  <fragment
4      android:id="@+id/landingFragment"
5      android:name="com.backbase.cse.retailapp.landing.LandingFragment"
6      android:label="LandingFragment" >
7  <action
8      android:id="@+id/action_landingFragment_to_authenticationJourney"
9      app:destination="@id/authenticationJourney" />

```

```

10     <action
11         android:id="@+id/action_landingFragment_to_onboardingScreen"
12         app:destination="@id/onboardingScreen" />
13 </fragment>
14
15 ...

```

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

```

1  import android.os.Bundle
2  import android.view.View
3  import androidx.fragment.app.Fragment
4  import androidx.navigation.fragment.findNavController
5  import com.backbase.cse.R
6  import kotlinx.android.synthetic.main.fragment_landing.btn_login
7  import kotlinx.android.synthetic.main.fragment_landing.btn_signup
8
9  class LandingFragment : Fragment(R.layout.fragment_landing) {
10
11      override fun onCreateView(view: View, savedInstanceState: Bundle?) {
12          super.onCreateView(view, savedInstanceState)
13
14          btn_login.setOnClickListener {
15              findNavController().navigate(R.id.action_landingFragment_to_authenticationJourney)
16          }
17
18          btn_signup.setOnClickListener {
19              findNavController().navigate(R.id.action_landingFragment_to_onboardingScreen)
20          }
21      }
22  }
23 }

```

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

```

1  {
2      "development": {
3          "debugEnable": false

```

```

4  },
5  "backbase": {
6      "serverURL": "<your-server-url>",
7      "localModelPath": "$(contextRoot)/localModel.json",
8      "experience": "",
9      "version": "6.1.5",
10     "identity": {
11         "baseURL": <your-identity-server-url>,
12         "realm": "retail",
13         "clientId": "mobile-client",
14         "applicationKey": "retail"
15     }
16 },
17 },
18 "security": {
19     "sslPinning": {
20         "checkChain": true
21     }
22 }
23 }
24

```

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

```

1  private fun initBackbase() {
2      initializeBackbase(
3          backbaseConfigAssetPath = "backbase/config.json", false
4      ).also {
5          setHttpHeaders()
6      }
7  }
8  }

```

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

```

1  import android.content.Context
2  import com.backbase.android.Backbase
3  import com.backbase.android.dbs.dataproviders.AssetsFileDBSDataProvider
4  import com.backbase.android.dbs.dataproviders.NetworkDBSDataProvider
5  import com.backbase.android.flow.aboutyou.AboutYouConfiguration
6  import com.backbase.android.flow.aboutyou.aboutYouJourneyModule
7  import com.backbase.android.flow.aboutyou.usecase.AboutYouUseCase
8  import com.backbase.android.flow.aboutyou.usecase.AboutYouUseCaseDefaultImpl
9  import com.backbase.android.flow.address.AddressConfiguration
10 import com.backbase.android.flow.address.addressJourneyModule
11 import com.backbase.android.flow.address.usecase.AddressUseCase
12 import com.backbase.android.flow.address.usecase.AddressUseCaseDefaultImpl
13 import com.backbase.android.flow.coapplicant.CoApplicantConfiguration
14 import com.backbase.android.flow.coapplicant.coApplicantJourneyModule
15 import com.backbase.android.flow.coapplicant.usecase.CoApplicantUseCase

```

```
16 import com.backbase.android.flow.coapplicant.usecase.CoApplicantUseCaseDefaultImpl
17 import com.backbase.android.flow.coapplicant.welcome.CoApplicantWelcomeConfiguration
18 import com.backbase.android.flow.coapplicant.welcome.coApplicantWelcomeJourneyModule
19 import com.backbase.android.flow.coapplicant.welcome.usecase.CoApplicantWelcomeUseCase
20 import com.backbase.android.flow.coapplicant.welcome.usecase.CoApplicantWelcomeUseCaseDefaultImpl
21 import com.backbase.android.flow.common.interaction.OnboardingBaseUseCase
22 import com.backbase.android.flow.common.interaction.OnboardingBaseUseCaseDefaultImpl
23 import com.backbase.android.flow.common.v2.interaction.OnboardingBaseUseCase as OnboardingBaseUseCase2
24 import com.backbase.android.flow.common.v2.interaction.OnboardingBaseUseCaseDefaultImpl as OnboardingBaseUs
25 import com.backbase.android.flow.contracts.FlowClientContract
26 import com.backbase.android.flow.v2.contracts.FlowClientContract as FlowClientContract2
27 import com.backbase.android.flow.v2.FlowClient as FlowClient2
28 import com.backbase.android.flow.FlowClient
29 import com.backbase.android.flow.common.model.OnboardingModel
30 import com.backbase.android.flow.common.utils.readAsset
31 import com.backbase.android.flow.credentials.CredentialsConfiguration
32 import com.backbase.android.flow.credentials.CredentialsJourneyModule
33 import com.backbase.android.flow.credentials.usecase.CredentialsUseCase
34 import com.backbase.android.flow.credentials.usecase.CredentialsUseCaseDefaultImpl
35 import com.backbase.android.flow.declined.DeclinedConfiguration
36 import com.backbase.android.flow.declined.declinedJourneyModule
37 import com.backbase.android.flow.declined.models.DeclinedModel
38 import com.backbase.android.flow.declined.usecase.DeclinedUseCase
39 import com.backbase.android.flow.declined.usecase.DeclinedUseCaseDefaultImpl
40 import com.backbase.android.flow.done.doneJourneyModule
41 import com.backbase.android.flow.done.usecase.DoneUseCase
42 import com.backbase.android.flow.done.usecase.DoneUseCaseDefaultImpl
43 import com.backbase.android.flow.identityverification.DocScannerDataCenter
44 import com.backbase.android.flow.identityverification.IdentityVerificationConfiguration
45 import com.backbase.android.flow.identityverification.IdentityVerificationJourneyModule
46 import com.backbase.android.flow.identityverification.usecase.IdentityVerificationUseCase
47 import com.backbase.android.flow.identityverification.usecase.IdentityVerificationUseCaseDefaultImpl
48 import com.backbase.android.flow.identityverified.IdentityVerifiedConfiguration
49 import com.backbase.android.flow.identityverified.identityVerifiedJourneyModule
50 import com.backbase.android.flow.identityverified.usecase.IdentityVerifiedUseCase
51 import com.backbase.android.flow.identityverified.usecase.IdentityVerifiedUseCaseDefaultImpl
52 import com.backbase.android.flow.inreview.inReviewJourneyModule
53 import com.backbase.android.flow.inreview.usecase.InReviewUseCase
54 import com.backbase.android.flow.inreview.usecase.InReviewUseCaseDefaultImpl
55 import com.backbase.android.flow.kyc.KycConfiguration
56 import com.backbase.android.flow.kyc.kycJourneyModule
57 import com.backbase.android.flow.kyc.usecase.KycUseCase
58 import com.backbase.android.flow.kyc.usecase.KycUseCaseDefaultImpl
59 import com.backbase.android.flow.loading.*
60 import com.backbase.android.flow.loading.usecase.LoadingUseCase
61 import com.backbase.android.flow.loading.usecase.LoadingUseCaseDefaultImpl
62 import com.backbase.android.flow.onboarding.OnboardingConfiguration
63 import com.backbase.cse.retailapp.OnboardingConstants.Companion.DBS_PATH
64 import com.backbase.cse.retailapp.OnboardingConstants.Companion.JUMIO_API_SECRET
65 import com.backbase.cse.retailapp.OnboardingConstants.Companion.JUMIO_API_TOKEN
66 import com.backbase.cse.retailapp.OnboardingConstants.Companion.ONBOARDING_INTERACTION_NAME
67 import com.backbase.cse.retailapp.OnboardingConstants.Companion.SERVICE_NAME
68 import com.backbase.cse.retailapp.OnboardingConstants.Companion.SHARED_CONTENT_REPOSITORY
69 import com.backbase.cse.retailapp.usecase.WelcomeJourneyUseCaseImpl
70 import com.backbase.cse.retailapp.walkthrough.securePage
71 import com.backbase.cse.retailapp.walkthrough.termsAndConditionsPages
72 import com.backbase.cse.retailapp.walkthrough.welcomePage
73 import com.backbase.android.flow.onboarding.models.Step
```

```

74 import com.backbase.android.flow.onboarding.onboardingModule
75 import com.backbase.android.flow.otp.OtpConfiguration
76 import com.backbase.android.flow.otp.otpJourneyModule
77 import com.backbase.android.flow.otp.usecase.OtpUseCase
78 import com.backbase.android.flow.otp.usecase.OtpUseCaseDefaultImpl
79 import com.backbase.android.flow.productselector.*
80 import com.backbase.android.flow.walkthrough.usecase.WalkthroughUseCase
81 import com.backbase.android.flow.walkthrough.usecase.WalkthroughUseCaseDefaultImpl
82 import com.backbase.android.flow.walkthrough.walkthroughJourneyModule
83 import com.backbase.android.flow.ssn.SsnConfiguration
84 import com.backbase.android.flow.ssn.ssnJourneyModule
85 import com.backbase.android.flow.ssn.usecase.SsnUseCase
86 import com.backbase.android.flow.ssn.usecase.SsnUseCaseDefaultImpl
87 import com.backbase.android.flow.walkthrough.WalkthroughConfiguration
88 import com.backbase.android.flow.welcome.WelcomeJourneyUseCase
89 import com.backbase.android.flow.welcome.welcomeJourneyModule
90 import com.backbase.android.flow.welcome.welcomeScreenConfiguration
91 import com.backbase.cse.R
92 import com.backbase.deferredresources.DeferredText
93 import com.google.gson.Gson
94 import com.google.gson.reflect.TypeToken
95 import org.koin.core.context.loadKoinModules
96 import org.koin.dsl.module
97 import java.lang.reflect.Type
98 import java.net.URI
99
100 val applicationModule = module {
101
102     single {
103         Gson()
104     }
105
106     single {
107         NetworkDBSDataProvider(get())
108     }
109
110     factory {
111         AssetsFileDBSDataProvider(get())
112     }
113
114     val serverURL = Backbase.getInstance()?.configuration?.experienceConfiguration?.serverURL
115     val baseURI = "$serverURL/$DBS_PATH/$SERVICE_NAME"
116     single<FlowClientContract>() {
117         FlowClient(
118             get(),
119             URI(baseURI),
120             get<NetworkDBSDataProvider>(),
121             null,
122             ONBOARDING_INTERACTION_NAME,
123             null
124         )
125     }
126     single<FlowClientContract2>() {
127         FlowClient2(
128             context = get(),
129             baseUri = URI(baseURI),
130             dataProvider = get<NetworkDBSDataProvider>(),
131             interactionName = ONBOARDING_INTERACTION_NAME,

```

```
132         interactionResponseDefaultBodyType = OnboardingModel::class.java
133     )
134 }
135
136 factory<OnboardingBaseUseCase2> {
137     OnboardingBaseUseCaseDefaultImpl2(get())
138 }
139
140 factory<OnboardingBaseUseCase> {
141     OnboardingBaseUseCaseDefaultImpl(get())
142 }
143 //endregion
144
145 // region Beneficiary Journey
146 factory<ProductSelectorUseCase>{
147     ProductSelectorUseCaseDefaultImpl(get(), get())
148 }
149
150 factory {
151     ProductSelectorConfiguration {
152         imageBaseUrl = serverURL.toString()
153         requestProductsAction = "get-product-list"
154         submitProductAction = "select-products"
155         createCaseAction = ""
156         selectionType = SelectionType.SINGLE
157     }
158 }
159
160 loadKoinModules(listOf(ProductSelectorJourneyModule))
161
162 //endregion
163
164 // region Co-Applicant Journey
165 factory<CoApplicantUseCase> {
166     CoApplicantUseCaseDefaultImpl(get(), get())
167 }
168
169 factory {
170     val context: Context by inject()
171     CoApplicantConfiguration {
172         coApplicantActionName = "account-type-selector"
173     }
174 }
175
176 loadKoinModules(listOf(coApplicantJourneyModule))
177
178 //endregion
179
180 // region Co-Applicant Welcome Journey
181 factory<CoApplicantWelcomeUseCase> {
182     CoApplicantWelcomeUseCaseDefaultImpl(get(), get())
183 }
184
185 factory {
186     val context: Context by inject()
187     CoApplicantWelcomeConfiguration {
188         coApplicantFetchDataActionName = "fetch-co-applicant-data"
189         coApplicantAttachActionName = "fetch-co-applicant"
```



```

190         coApplicantWelcomeInteractionName = "co-applicant-welcome"
191         onboardingInteractionName = ONBOARDING_INTERACTION_NAME
192     }
193 }
194
195 loadKoinModules(listOf(coApplicantWelcomeJourneyModule))
196
197 //endregion
198
199 //region Loading Journey
200 factory<LoadingUseCase> {
201     LoadingUseCaseDefaultImpl(get(), get())
202 }
203
204 factory {
205     LoadingConfiguration {
206         initializeActionName = "fetch-co-applicant"
207     }
208 }
209 loadKoinModules(listOf(loadingJourneyModule))
210 //endregion
211
212 // region Welcome Journey
213 factory<WelcomeJourneyUseCase> {
214     WelcomeJourneyUseCaseImpl()
215 }
216
217 factory {
218     val context: Context by inject()
219     welcomeScreenConfiguration {
220         brandTitle = DeferredText.Constant(context.getString(R.string.brand_title))
221         brandTitleDetail = DeferredText.Constant(context.getString(R.string.brand_title_detail))
222     }
223 }
224
225 loadKoinModules(listOf(welcomeJourneyModule))
226
227 //endregion
228
229 //region Walkthrough journey
230
231 factory {
232     walkthroughConfiguration {
233         pages = arrayListOf(
234             welcomePage(),
235             securePage(get()),
236             termsAndConditionsPages()
237         )
238         privacyPolicyUrl = "https://www.backbase.com/privacy-policy/"
239         termsAndConditionsUrl = "https://www.backbase.com/legal/"
240         submitTermsAndConditionsActionName = "agree-to-terms"
241         agreedToTermsPayloadKey = "agreedToTerms"
242     }
243 }
244
245 factory<WalkthroughUseCase> {
246     WalkthroughUseCaseDefaultImpl(get(), get())
247 }

```

```

248
249     loadKoinModules(listOf(walkthroughJourneyModule))
250
251     //endregion
252
253     //region onboarding journey
254     factory {
255         AboutYouConfiguration {
256             val gson: Gson by inject()
257             val context: Context by inject()
258             val formItemsType: Type = object :
259                 TypeToken<ArrayList<com.backbase.android.flow.aboutyou.models.FormItem>>() {}.type
260             formItems = gson.fromJson(
261                 readAsset(
262                     context.assets,
263                     "backbase/onboarding/anchor.json"
264                 ), formItemsType
265             )
266             submitActionName = "submit-anchor-data"
267             prefillActionName = "prefill-anchor-data"
268         }
269     }
270
271     factory<AboutYouUseCase> {
272         AboutYouUseCaseDefaultImpl(get(), get())
273     }
274
275     loadKoinModules(listOf(aboutYouJourneyModule))
276     //endregion
277
278     //region ssn journey
279     factory {
280         SsnConfiguration {
281             val gson: Gson by inject()
282             val context: Context by inject()
283             val formItemsType: Type = object :
284                 TypeToken<ArrayList<com.backbase.android.flow.ssn.models.FormItem>>() {}.type
285             formItems = gson.fromJson(
286                 readAsset(
287                     context.assets,
288                     "backbase/onboarding/ssn.json"
289                 ), formItemsType
290             )
291             submitSsnActionName = "submit-ssn"
292             selectCitizenshipActionName = "citizenship-selector"
293         }
294     }
295
296     factory<SsnUseCase> {
297         SsnUseCaseDefaultImpl(get(), get())
298     }
299
300     loadKoinModules(listOf(ssnJourneyModule))
301     //endregion
302
303     //region onboarding journey
304
305     factory {

```

```

306         OnboardingConfiguration {
307             this.steps = steps
308         }
309     }
310
311     loadKoinModules(listOf(onboardingModule))
312
313     //endregion
314
315     //region Address Validation Journey
316     factory {
317         AddressConfiguration {
318             submitActionName = "submit-address"
319             fetchActionName = "fetch-address"
320         }
321     }
322
323     factory<AddressUseCase> {
324         AddressUseCaseDefaultImpl(get(), get())
325     }
326
327     loadKoinModules(listOf(addressJourneyModule))
328     //endregion
329
330     //region OTP
331
332     factory {
333         OtpConfiguration {
334             requestActionName = "request-otp"
335             verifyActionName = "verify-otp"
336             availableOtpChannelsActionName = "available-otp-channels"
337             verificationCodeMaxLength = Integer(6)
338         }
339     }
340
341     factory<OtpUseCase> {
342         OtpUseCaseDefaultImpl(get(), get())
343     }
344
345     loadKoinModules(listOf(otpJourneyModule))
346     //endregion
347
348     //region
349     loadKoinModules(listOf(CredentialsJourneyModule))
350
351     factory {
352         CredentialsConfiguration {
353             actionName = "setup-credentials"
354         }
355     }
356
357     factory<CredentialsUseCase> {
358         CredentialsUseCaseDefaultImpl(get(), get())
359     }
360     //endregion
361
362     //region identity_verified
363     factory {

```

```

364     IdentityVerifiedConfiguration {
365         actionName = "empty-handler"
366     }
367 }
368
369 factory<IdentityVerifiedUseCase> {
370     IdentityVerifiedUseCaseDefaultImpl(get(), get())
371 }
372
373 loadKoinModules(listOf(identityVerifiedJourneyModule))
374 //endregion
375
376 //region successfully-done
377 factory<DoneUseCase> {
378     DoneUseCaseDefaultImpl(get())
379 }
380 loadKoinModules(listOf(doneJourneyModule))
381 //endregion
382
383 //region in-review-done
384 factory<InReviewUseCase> {
385     InReviewUseCaseDefaultImpl(get())
386 }
387 loadKoinModules(listOf(inReviewJourneyModule))
388 //endregion
389
390 //region declined
391
392 factory {
393     DeclinedConfiguration {
394         declinedFileName = "/declinedScreen.json"
395         contentRepoName = SHARED_CONTENT_REPOSITORY
396         declinedDataModel = DeclinedModel(
397             "http://www.backbase.com",
398             "http://www.backbase.com",
399             "http://www.backbase.com"
400         )
401     }
402 }
403
404 factory<DeclinedUseCase> {
405     DeclinedUseCaseDefaultImpl(
406         get(),
407         get(),
408         true
409     )
410 }
411
412 loadKoinModules(listOf(declinedJourneyModule))
413
414 //endregion
415
416 //region kyc
417
418 loadKoinModules(listOf(kycJourneyModule))
419
420 factory {
421     KycConfiguration {

```

```

422         requestKycQuestionsActionName = "kyc-questions-loader"
423         submitKycActionName = "submit-kyc-data"
424     }
425 }
426
427 factory<KycUsecase> {
428     KycUsecaseDefaultImpl(get(), get())
429 }
430 //endregion kyc
431
432 //region IDV
433 loadKoinModules(listOf(IdentityVerificationJourneyModule))
434
435 factory {
436     IdentityVerificationConfiguration {
437         apiToken = JUMIO_API_TOKEN
438         apiSecretKey = JUMIO_API_SECRET
439         dataCenter = DocScannerDataCenter.EU
440         initiationActionName = "identity-verification-initiation"
441         verificationActionName = "identity-verification-result"
442     }
443 }
444
445 factory<IdentityVerificationUseCase> {
446     IdentityVerificationUseCaseDefaultImpl(get(), get())
447 }
448 //endregion IDV
449 }
450
451
452

```

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

▼ MyApplication.kt

```

1  ...
2
3  override fun onCreate() {
4      super.onCreate()
5      initBackbase()
6
7      koinApplication {
8          loadKoinModules(applicationModule)
9      }
10 }
11
12 ...

```

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

```

1  import android.content.Context
2  import androidx.navigation.NavController

```

```

3 import com.backbase.android.flow.FlowClient
4 import com.backbase.android.flow.common.model.ApplicantType
5 import com.backbase.android.flow.common.model.StepConfiguration
6 import com.backbase.android.flow.contracts.FlowClientContract
7 import com.backbase.android.flow.listeners.InteractionListener
8 import com.backbase.android.flow.onboarding.OnboardingRouter
9 import com.backbase.android.utils.net.response.Response
10 import com.backbase.cse.R
11 import kotlinx.coroutines.GlobalScope
12 import kotlinx.coroutines.launch
13 import kotlinx.coroutines.runBlocking
14 import org.koin.java.KoinJavaComponent.inject
15 import kotlin.coroutines.resume
16 import kotlin.coroutines.suspendCoroutine
17
18 fun onboardingRouter(
19     navController: NavController,
20     usecase: FlowClientContract,
21     context: Context,
22     completion: () -> Unit = {}
23 ) = object : OnboardingRouter {
24     override fun onClose() {
25         resetInteractions(context)
26         showJourneyWithClearStack(
27             navController,
28             R.id.onboardingScreen
29         )
30         completion()
31     }
32
33     private fun resetInteractions(context: Context) {
34         val flowClient by inject(FlowClientContract::class.java)
35         (flowClient as FlowClient)?.let {
36             it.interactionId = null
37             StepConfiguration.model = null
38             StepConfiguration.interactionId = null
39             StepConfiguration.stepName = null
40             StepConfiguration.steps = null
41             StepConfiguration.previousStepName = null
42             ApplicantType.reset(context)
43         }
44     }
45
46     fun isBackNavigationSupported(): Pair<Boolean, String?> {
47         StepConfiguration.steps?.get(StepConfiguration.stepName)?.let { step ->
48             step.back?.let { back ->
49                 return Pair(step.enabled, back)
50             }
51         }
52         return Pair(false, null)
53     }
54
55     override fun goBack(): Boolean {
56         val (isEnabled, back) = isBackNavigationSupported()
57         if (isEnabled) {
58             back?.let {
59                 StepConfiguration.stepName = back
60                 StepConfiguration.previousStepName = StepConfiguration.steps?.get(back)?.back

```

```

61         return runBlocking {
62             suspendCoroutine<Boolean> {
63                 GlobalScope.launch {
64                     usecase.navigateToStep(back, object : InteractionListener<Object> {
65                         override fun onSuccess(o: Object) {
66                             it.resume(true)
67                         }
68
69                         override fun onError(response: Response) {
70                             it.resume(response.responseCode == 200)
71                         }
72                     })
73                 }
74             }
75         }
76     }
77 }
78
79     return false
80 }
81 }
82
83 fun showJourneyWithClearStack(
84     navController: NavController,
85     journeyScreenResId: Int
86 ) {
87     val graph = navController.graph
88     graph.startDestination = journeyScreenResId
89     navController.graph = graph
90 }
91

```

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

▼ MyAppActivity.kt

```

1  ...
2  private fun instantiateActivityModule() = module {
3      factory {
4          onboardingRouter(controller, get(), get()) {
5              setTheme(R.style.AppTheme)
6          }
7      }
8  }
9  private val activityModule: Module by lazy { instantiateActivityModule() }
10
11  override fun onCreate(savedInstanceState: Bundle?) {
12      super.onCreate(savedInstanceState)
13      loadKoinModules(activityModule)
14  }
15  ...

```

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.

