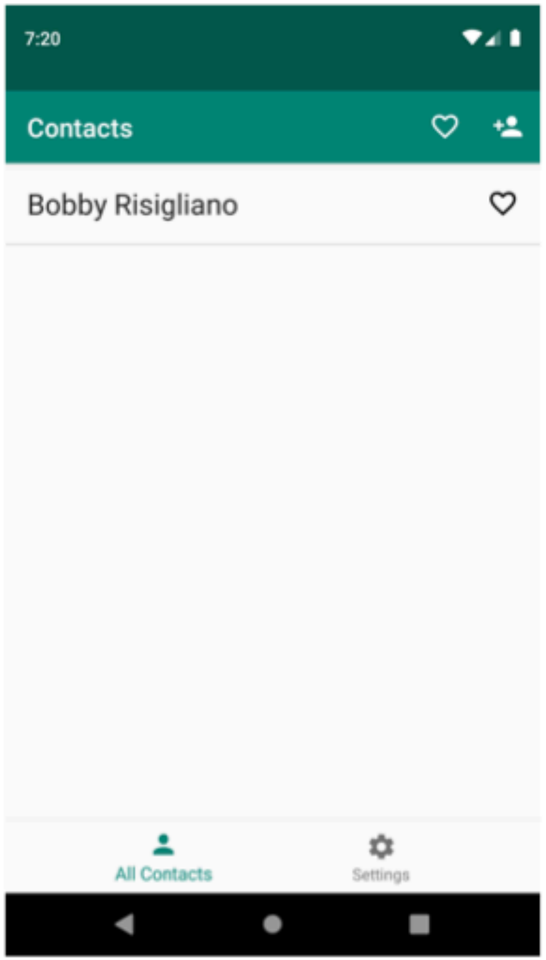
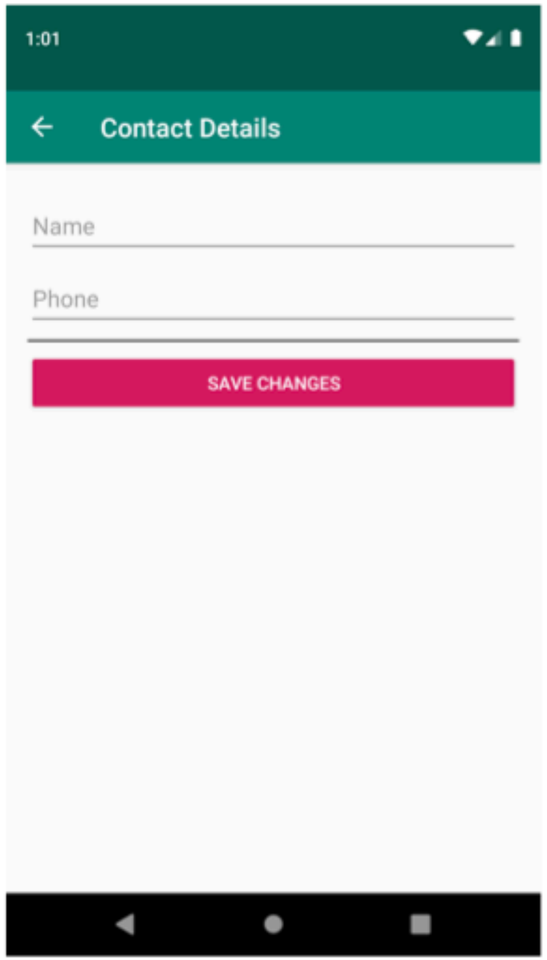
Navigation Workshop

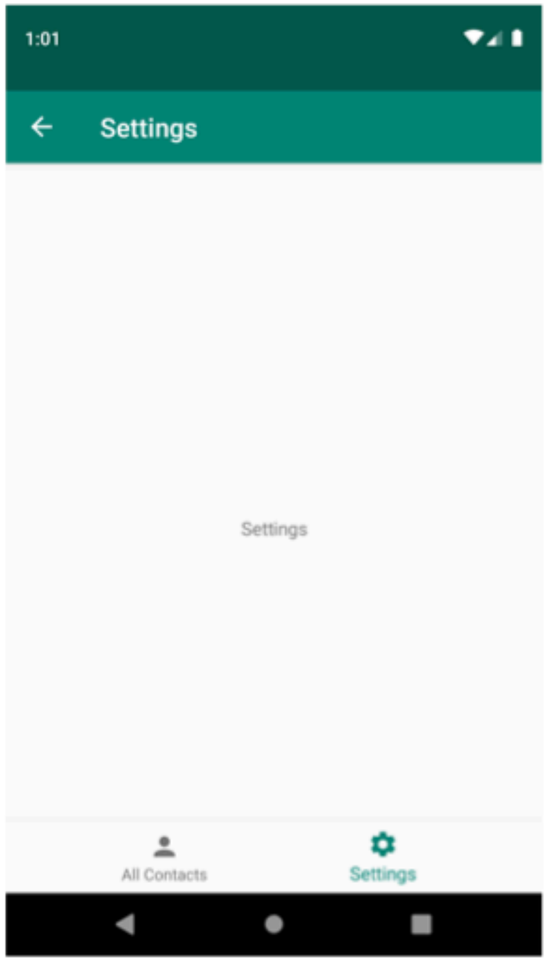
Eric Maxwell
@emmax

AGENDA

What to expect







AGENDA

What to expect

- Lab 1 - Getting Started
- Lab 2 - Top Level Navigation + Testing
- Lab 3 - Conditional Navigation + Deep Linking

AGENDA

What to expect

Requirements

- Android Studio 3.2.1 or Higher
- Emulator running API 28 or higher

Materials

- Lab manual : https://cutt.ly/as-2019-nav-lab

AGENDA

What to expect

Requirements

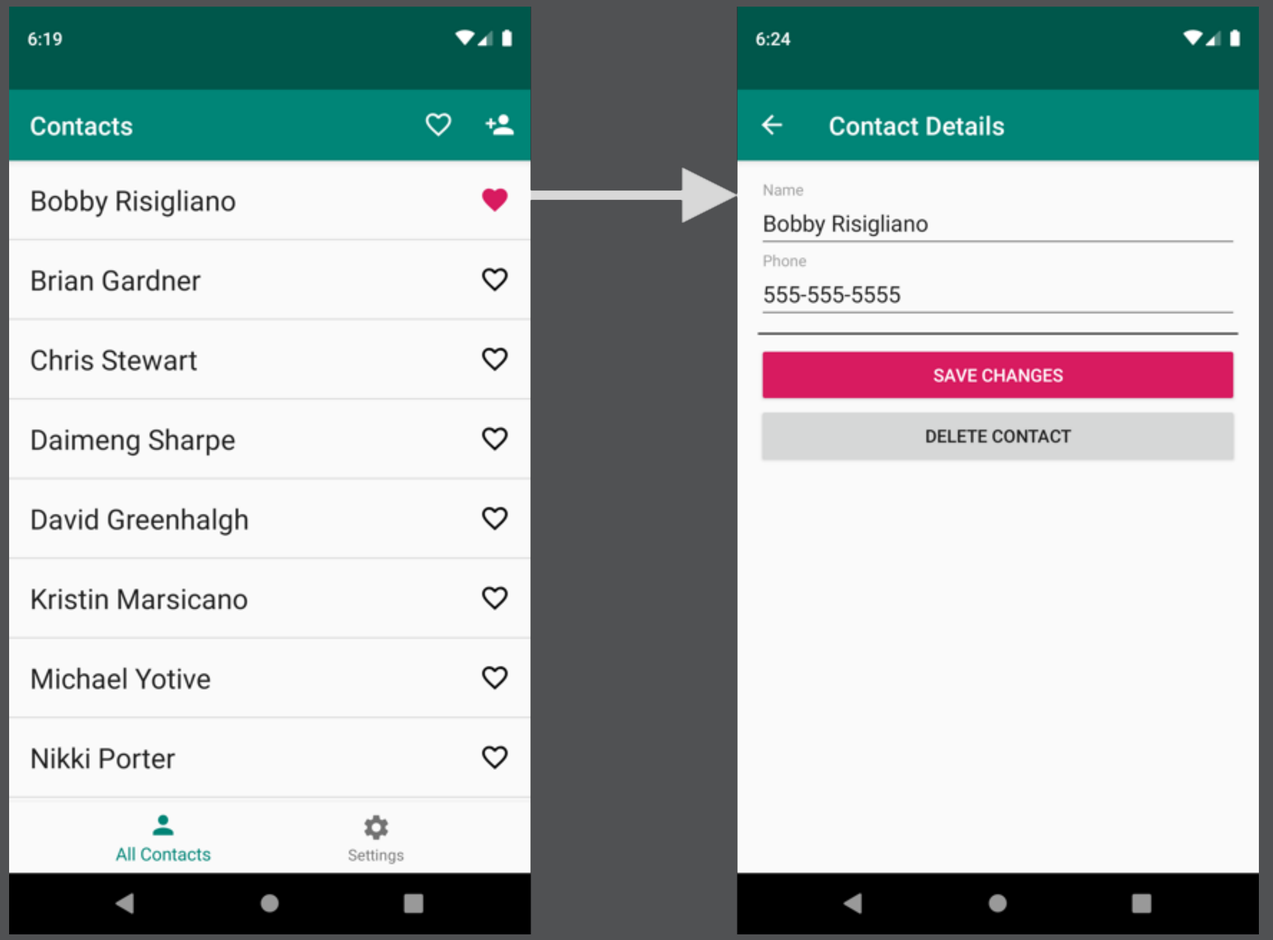
- Android Studio 3.2.1 or Higher
- Emulator running API 28 or higher

Materials

- Lab manual : https://cutt.ly/as-2019-nav-lab
  (Suggest ***uncheck "View* -> Print layout"**)

Lab 1: Getting Started

Motivations


6:19
Contacts
Bobby Risigliano
Brian Gardner
Chris Stewart
Daimeng Sharpe
David Greenhalgh
Kristin Marsicano
Michael Yotive
Nikki Porter
All Contacts
Settings
6:24
Contact Details
Name
Bobby Risigliano
Phone
555-555-5555
SAVE CHANGES
DELETE CONTACT

Pre Navigation

Navigate to Activity

```
val intent = Intent(this,
   ContactDetailActivity::class.java)

intent.putExtra("contactId", contact.id)

startActivity(intent)
```

Navigate to Fragment

```
val contactDetailFragment =
ContactDetailFragment().apply {
    arguments = Bundle().apply {
        putString("contactId", contactId)
    }
}

supportFragmentManager
    .beginTransaction()
    .replace(R.id.nav_host,
            contactDetailFragment)
    .addToBackStack(null)
    .commit()
```

```
// .. inside ContactDetailActivity

val contactId = intent
    .getStringExtra("contactId")
```

```
// ... inside ContactDetailFragment

val contactId = requireArguments()
    .getString("contactId")
```

Pre Navigation

Navigate to Activity

```
val intent = Intent(this,
   ContactDetailActivity::class.java)

intent.putExtra("contactId", contact.id)

startActivity(intent)
```

Navigate to Fragment

```
val contactDetailFragment = ContactDetailFragment().apply {
    arguments = Bundle().apply {
        putString("contactId", contactId)
    }
}

supportFragmentManager
    .beginTransaction()
    .replace(R.id.nav_host,
           contactDetailFragment)
    .addToBackStack(null)
    .commit()
```

```
// .. inside ContactDetailActivity

val contactId = intent
    .getStringExtra("contactId")
```

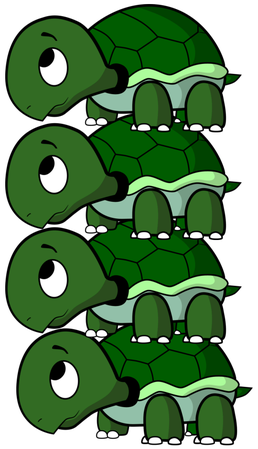
```
// ... inside ContactDetailFragment

val contactId = requireArguments()
    .getString("contactId")
```

Navigation

```
findNavController()
    .navigate(toContactDetail(contact.id))
```

Navigate to Activity

```
val intent = Intent(this,

ContactDetailActivity::class.java)

intent.putExtra("contactId",
contact.id)

startActivity(intent)

// .. inside ContactDetailFragment

val contactId = intent
    .getStringExtra("contactId")
```



Navigate to Fragment

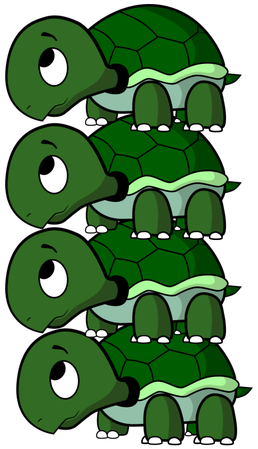
```
val contactDetailFragment = ContactDetailFragment().apply {
    arguments = Bundle().apply {
        putString("contactId", contactId)
    }
}

supportFragmentManager
    .beginTransaction()
    .replace(R.id.nav_host, contactDetailFragment)
    .addToBackStack(null)
    .commit()

// ... inside ContactDetailFragment

val contactId = requireArguments()
    .getString("contactId")
```

Navigation

```
val args = ContactDetailFragmentArgs
                    .fromBundle(requireArguments())

val contactId = args.contactId
```

Navigate to Activity

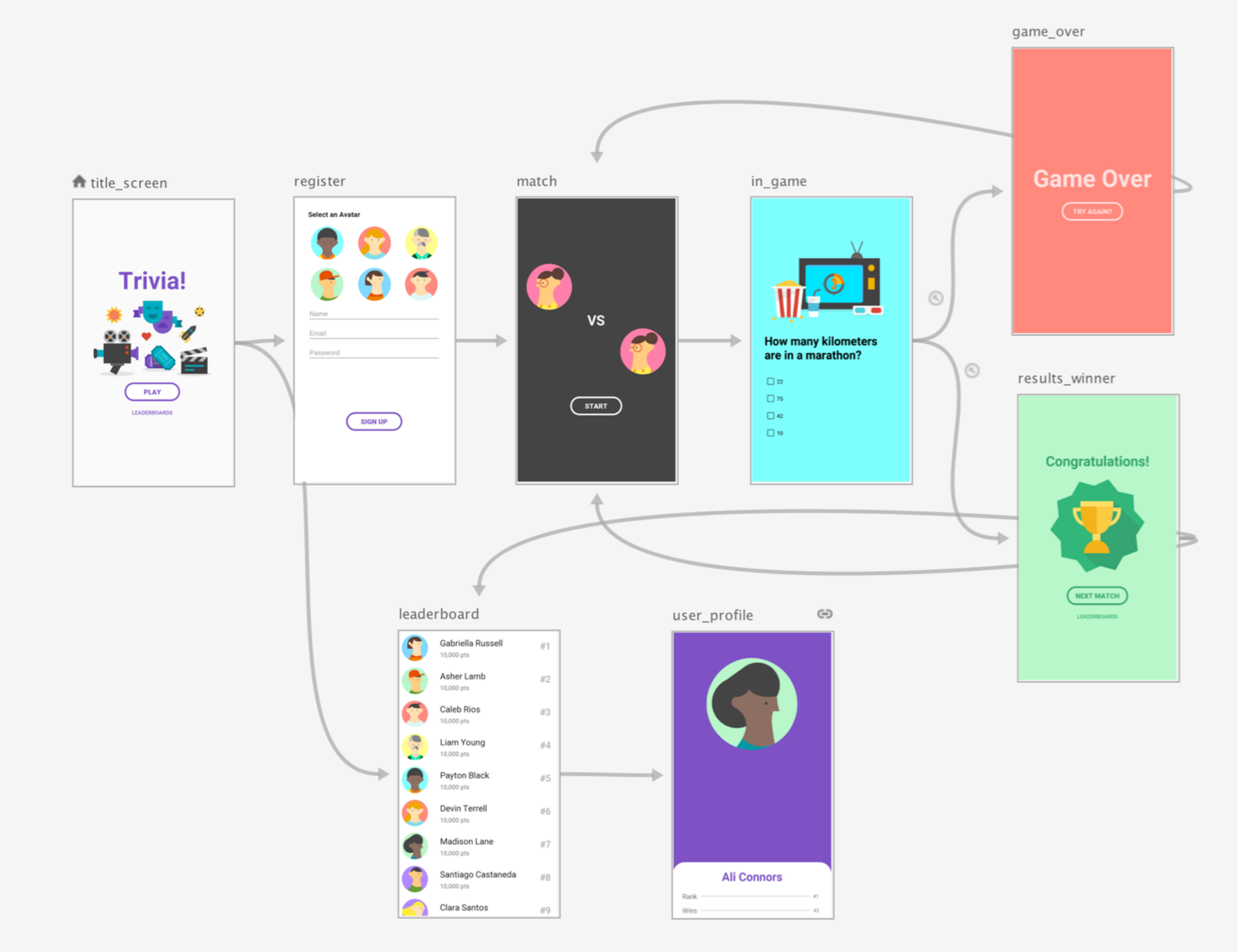
```
val intent = Intent(this,

ContactDetailActivity::class.java)

intent.putExtra("contactId",
contact.id)

startActivity(intent)

// .. inside ContactDetailFragment

val contactId = intent
    .getStringExtra("contactId")
```



Navigate to Fragment

```
val contactDetailFragment = ContactDetailFragment().apply {
    arguments = Bundle().apply {
        putString("contactId", contactId)
    }
}

supportFragmentManager
    .beginTransaction()
    .replace(R.id.nav_host, contactDetailFragment)
    .addToBackStack(null)
    .commit()

// ... inside ContactDetailFragment

val contactId = requireArguments()
    .getString("contactId")
```

How Does it Work?



https://github.com/googlesamples/android-architecture-components/tree/master/NavigationBasicSample

Minimal Dependencies

Navigation

```
dependencies {
    ...
    implementation "androidx.navigation:navigation-runtime-ktx:$version" /* Navigation Runtime */
    implementation "androidx.navigation:navigation-fragment-ktx:$version" /* Fragment Destination */

}
```

Minimal Dependencies

Navigation

```
dependencies {
    ...
    implementation "androidx.navigation:navigation-runtime-ktx:$version" /* Navigation Runtime */
    implementation "androidx.navigation:navigation-fragment-ktx:$version" /* Fragment Destination */
    implementation “androidx.navigation:navigation-ui-ktx:$version" /* Top-Level Navigation */
}
```

Minimal Dependencies

Navigation

```
dependencies {
    ...
    implementation "androidx.navigation:navigation-runtime-ktx:$version" /* Navigation Runtime */
    implementation "androidx.navigation:navigation-fragment-ktx:$version" /* Fragment Destination */
    implementation “androidx.navigation:navigation-ui-ktx:$version" /* Top-Level Navigation */
}
```

Safe Args

Project build.gradle:

```
dependencies {
   ...
   classpath “androidx.navigation:navigation-safe-args-gradle-plugin:$version"
}
```

Module build.gradle:

```
apply plugin: 'androidx.navigation.safeargs.kotlin'
```

Minimal Dependencies

Navigation

```
dependencies {
    ...
    implementation "androidx.navigation:navigation-runtime-ktx:$version" /* Navigation Runtime */
    implementation "androidx.navigation:navigation-fragment-ktx:$version" /* Fragment Destination */
    implementation “androidx.navigation:navigation-ui-ktx:$version" /* Top-Level Navigation */
}
```

Safe Args

Project build.gradle:

```
dependencies {
   ...
   classpath “androidx.navigation:navigation-safe-args-gradle-plugin:$version"
}
```

Module build.gradle:

```
apply plugin: 'androidx.navigation.safeargs.kotlin'
```

Minimal Dependencies

Navigation

```
dependencies {
    ...
    implementation "androidx.navigation:navigation-fragment-ktx:$version" /* Fragment Destination */
    implementation “androidx.navigation:navigation-ui-ktx:$version" /* Top-Level Navigation */
}
```

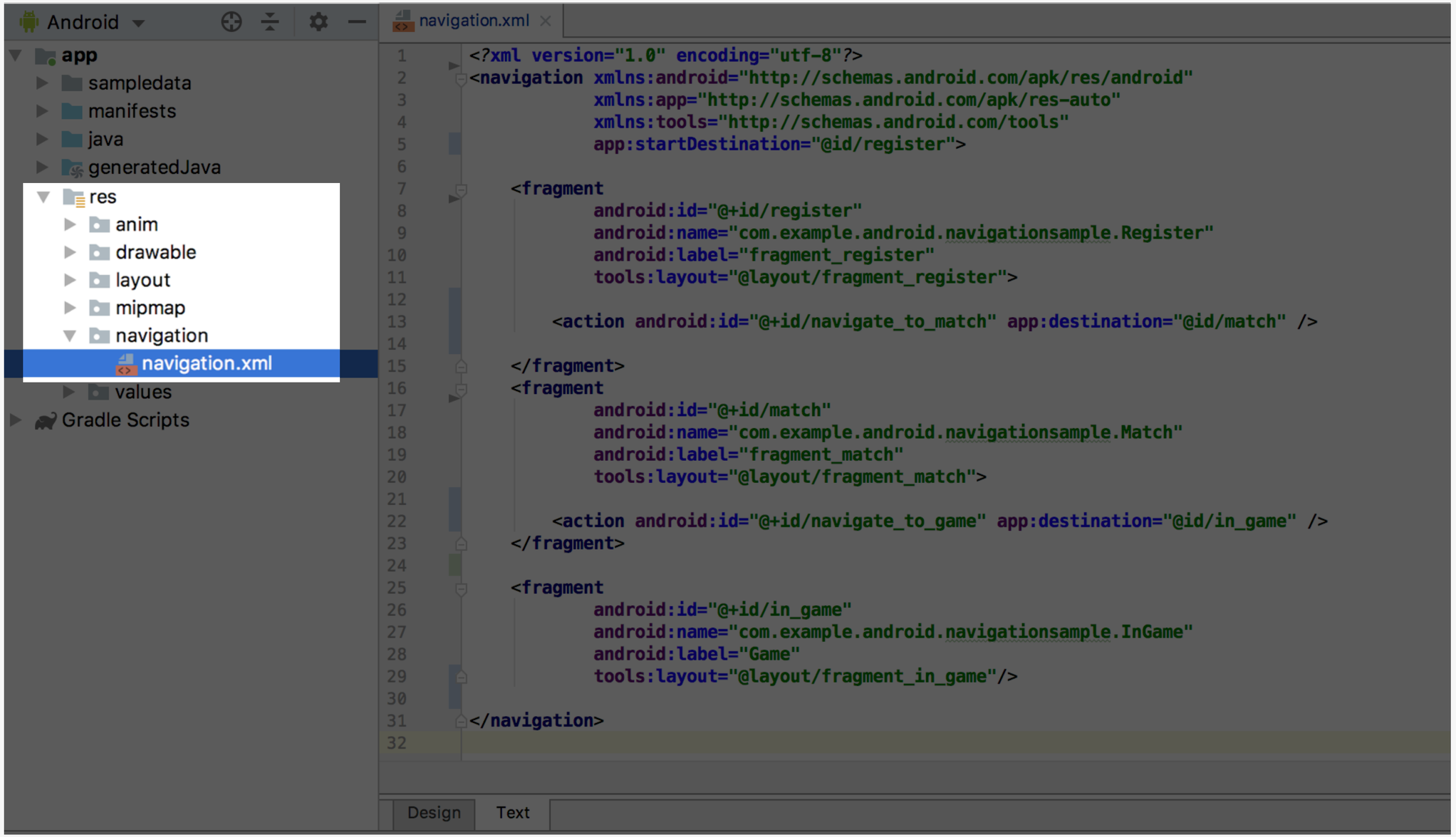
Safe Args

Project build.gradle:

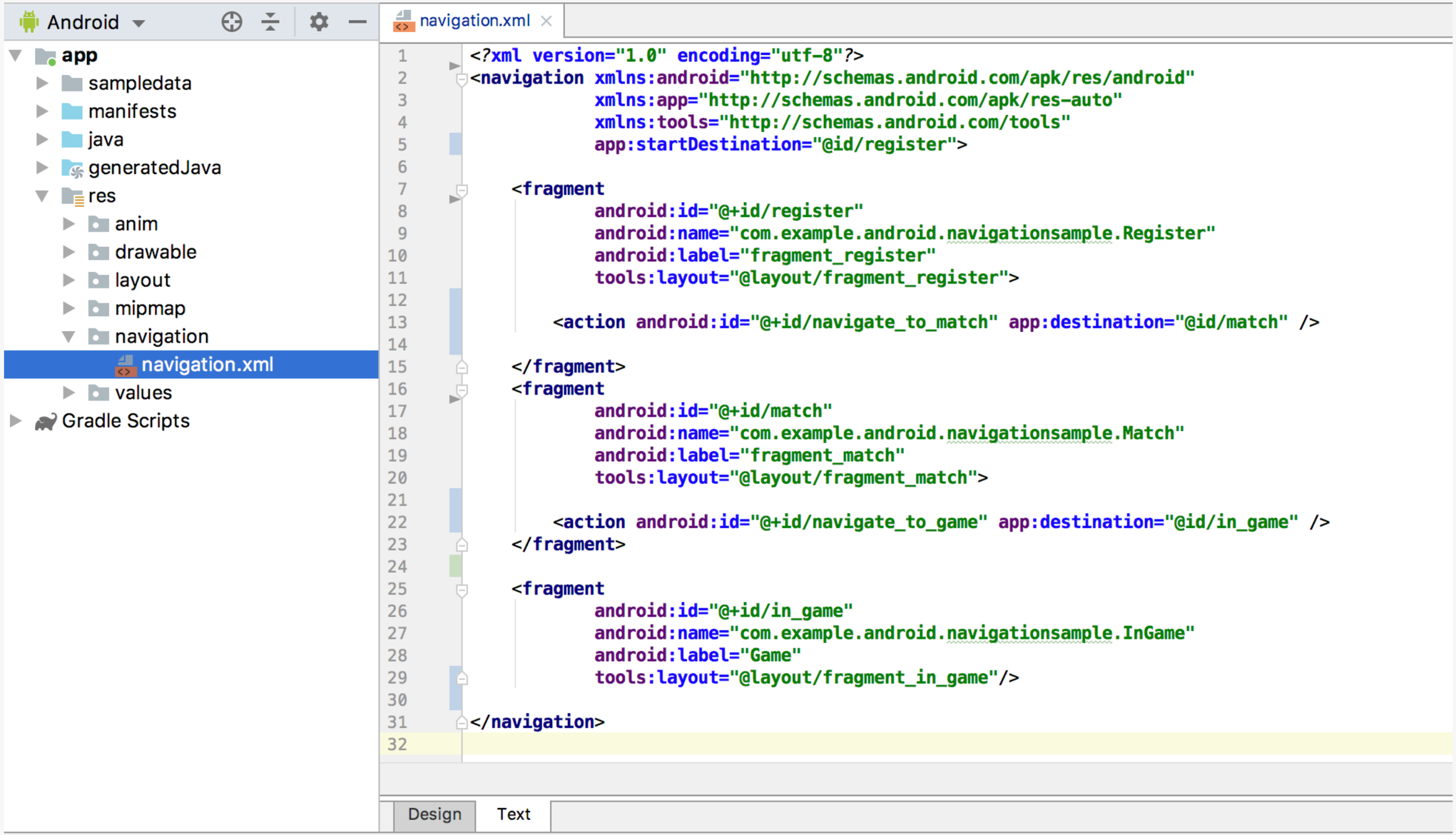
```
dependencies {
   ...
   classpath “androidx.navigation:navigation-safe-args-gradle-plugin:$version"
}
```

Module build.gradle:

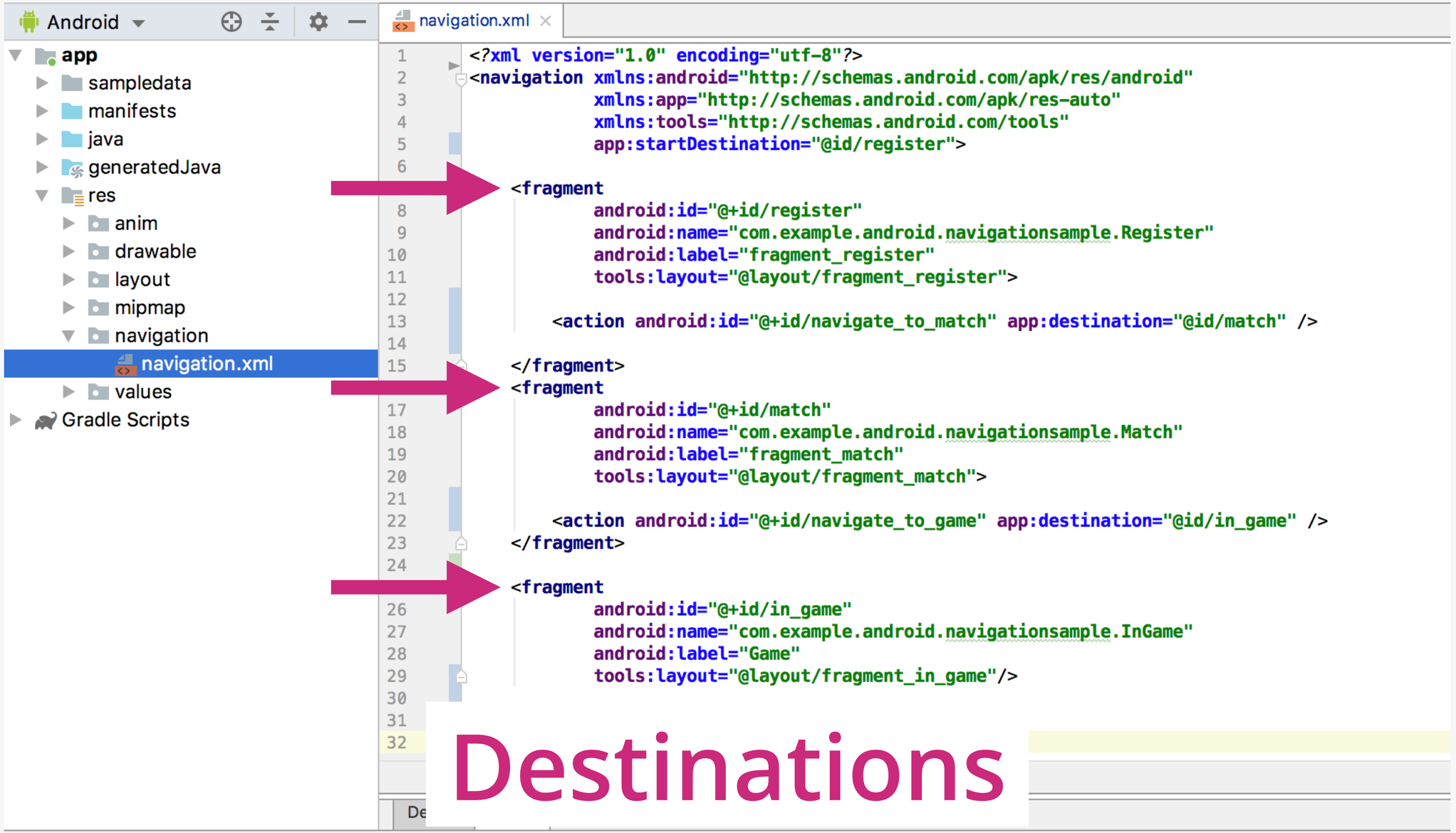
```
apply plugin: 'androidx.navigation.safeargs.kotlin'
```



https://github.com/googlesamples/android-architecture-components/tree/master/NavigationBasicSample

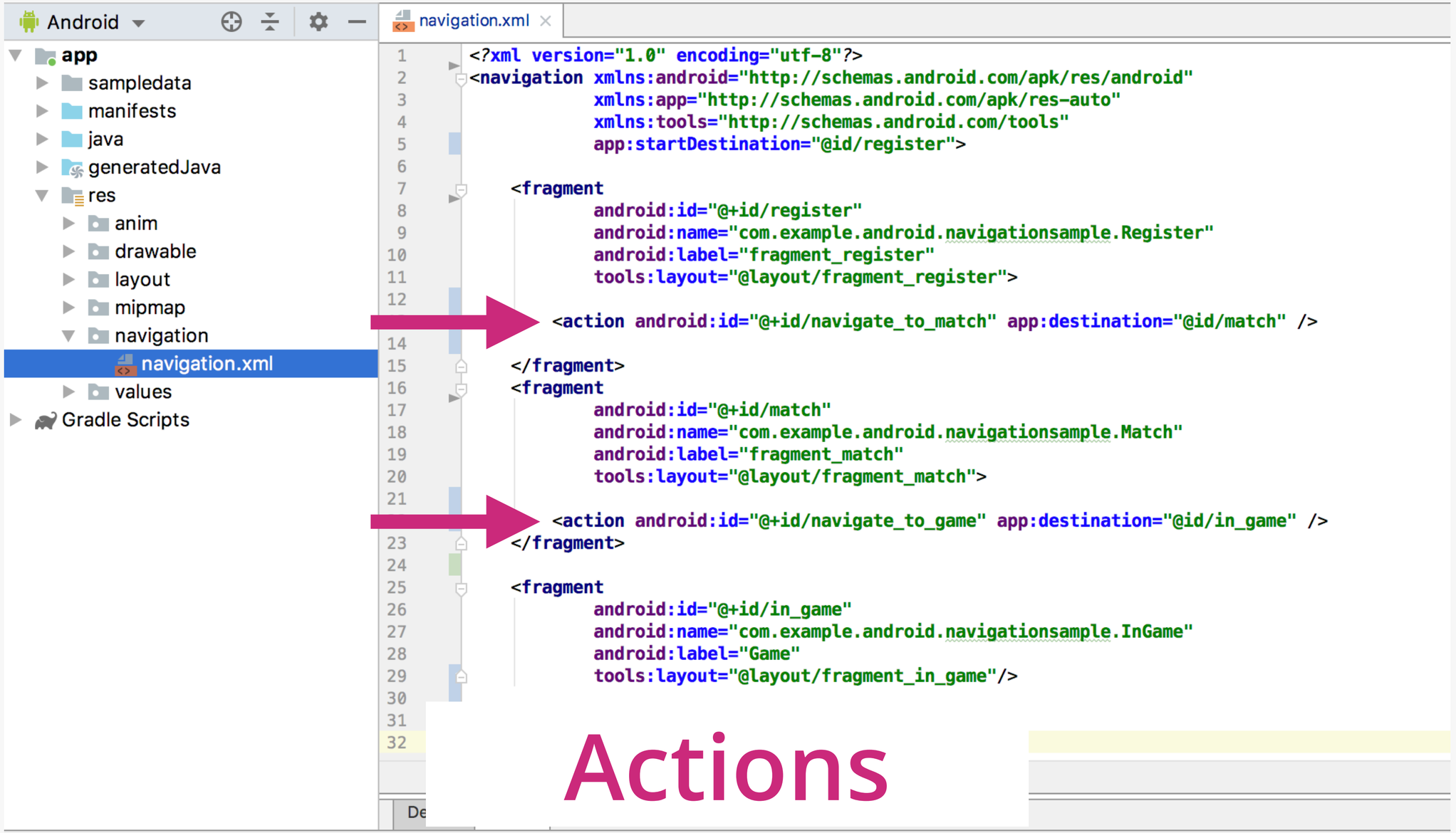


https://github.com/googlesamples/android-architecture-components/tree/master/NavigationBasicSample

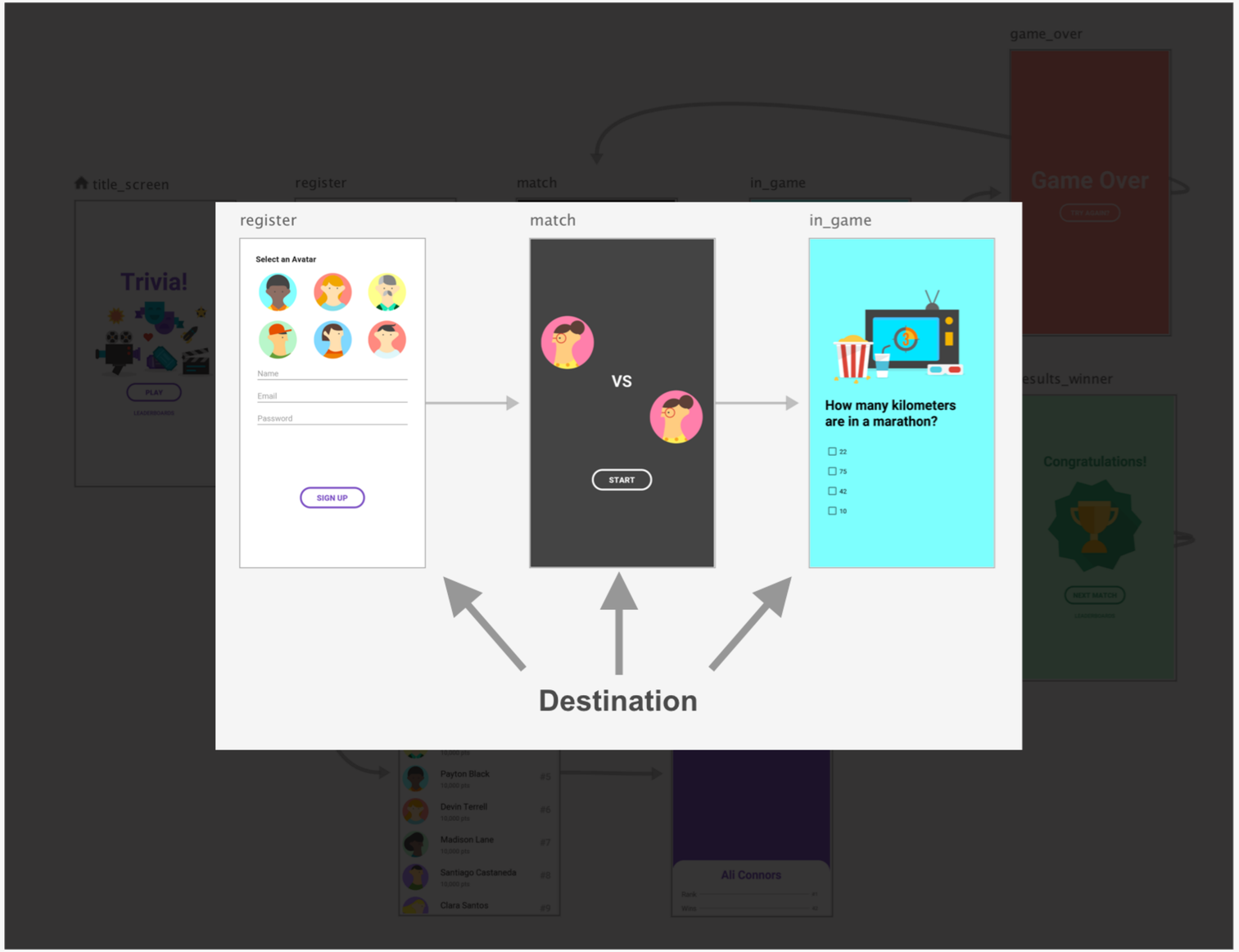


Destinations

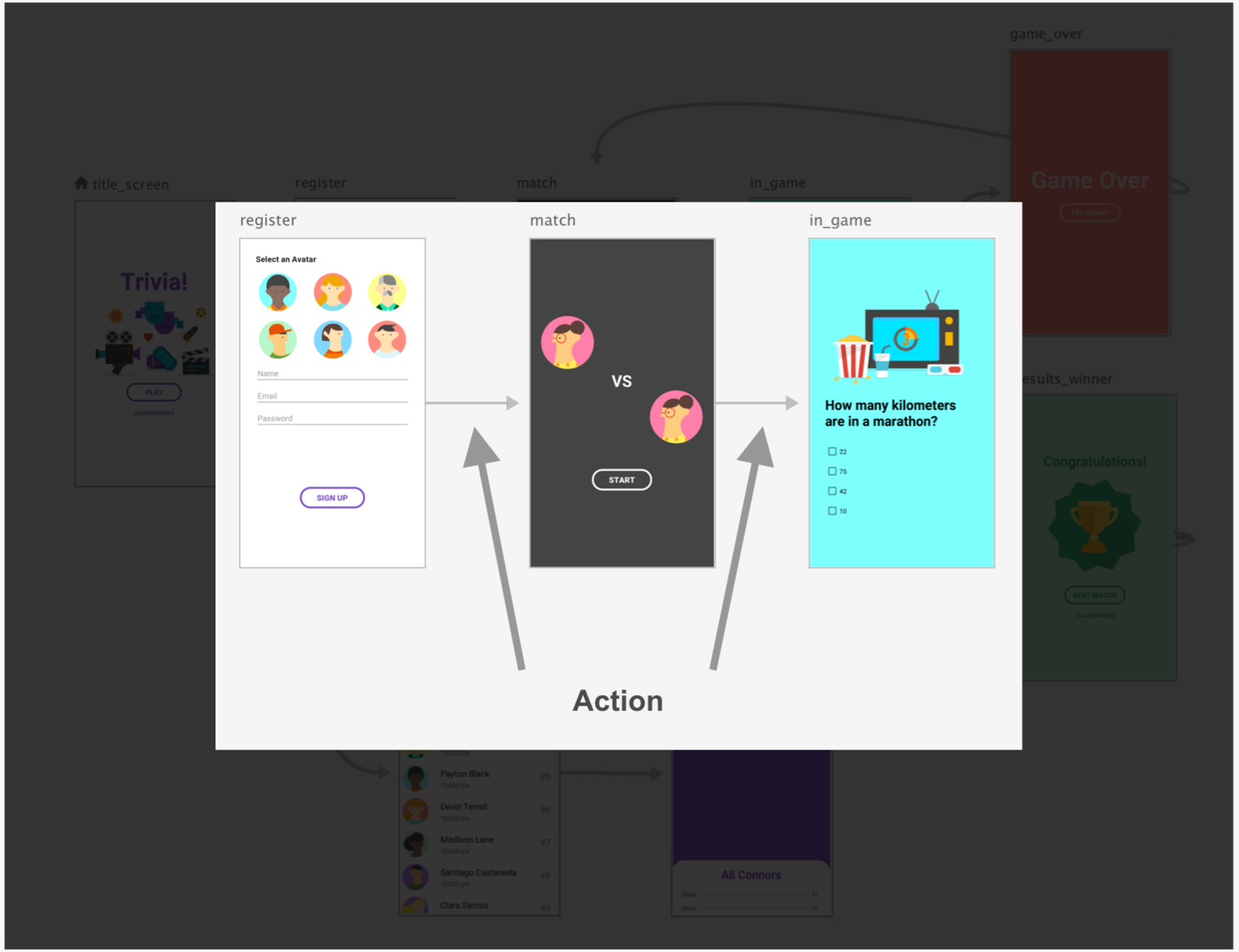
https://github.com/googlesamples/android-architecture-components/tree/master/NavigationBasicSample



https://github.com/googlesamples/android-architecture-components/tree/master/NavigationBasicSample



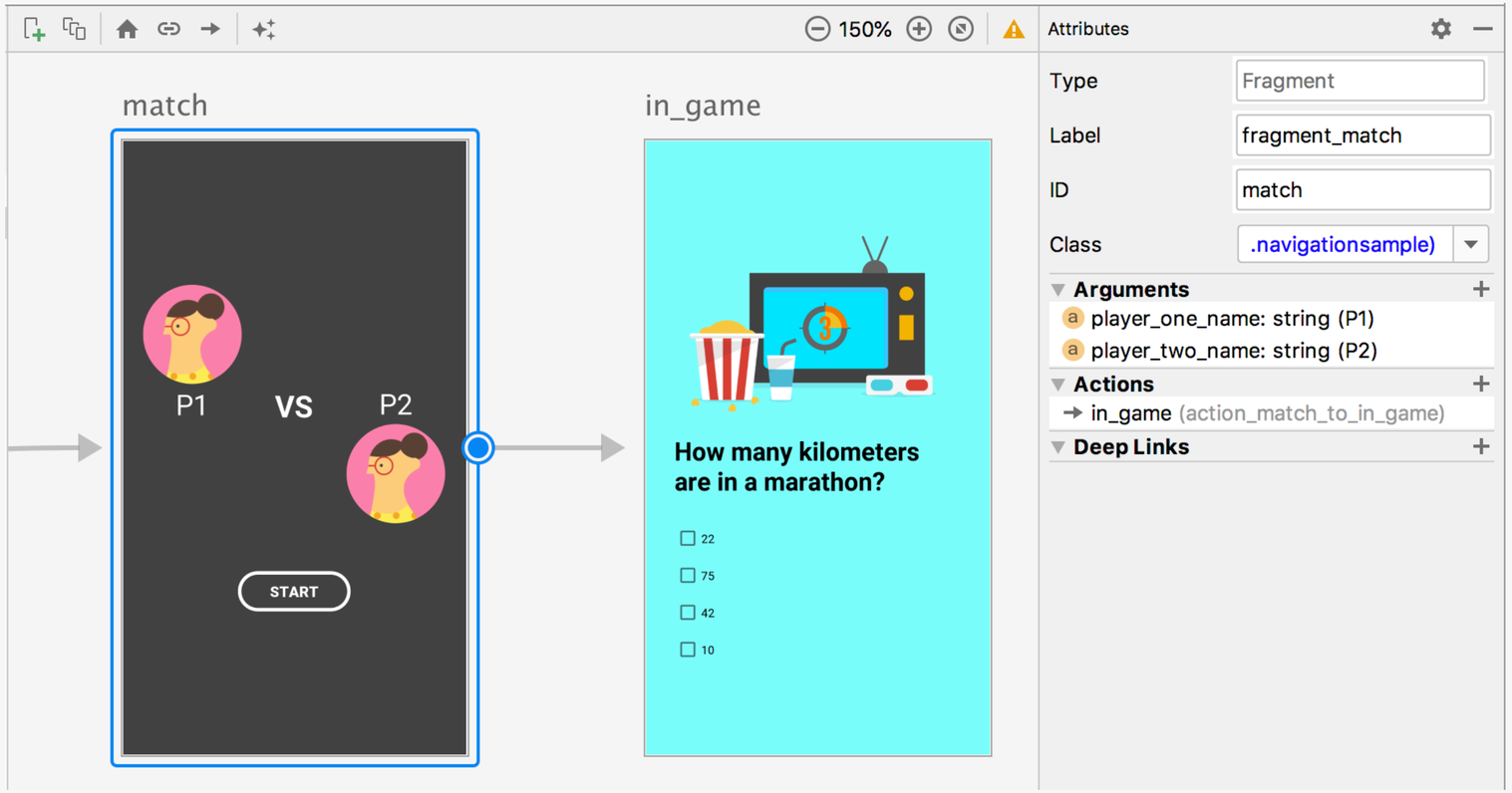
https://github.com/googlesamples/android-architecture-components/tree/master/NavigationBasicSample



https://github.com/googlesamples/android-architecture-components/tree/master/NavigationBasicSample

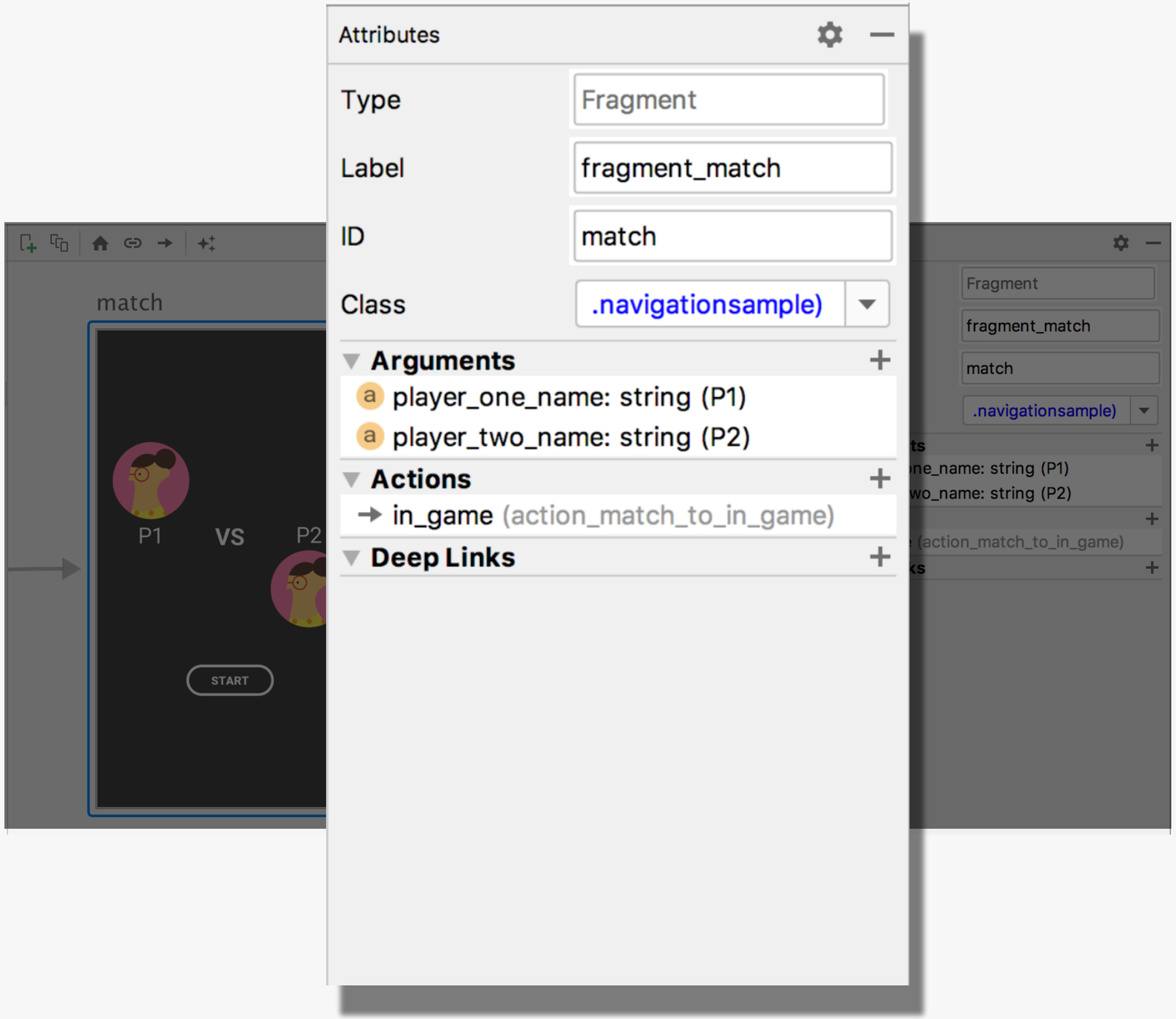
Destinations

- Arguments
- Deep Links
- Actions



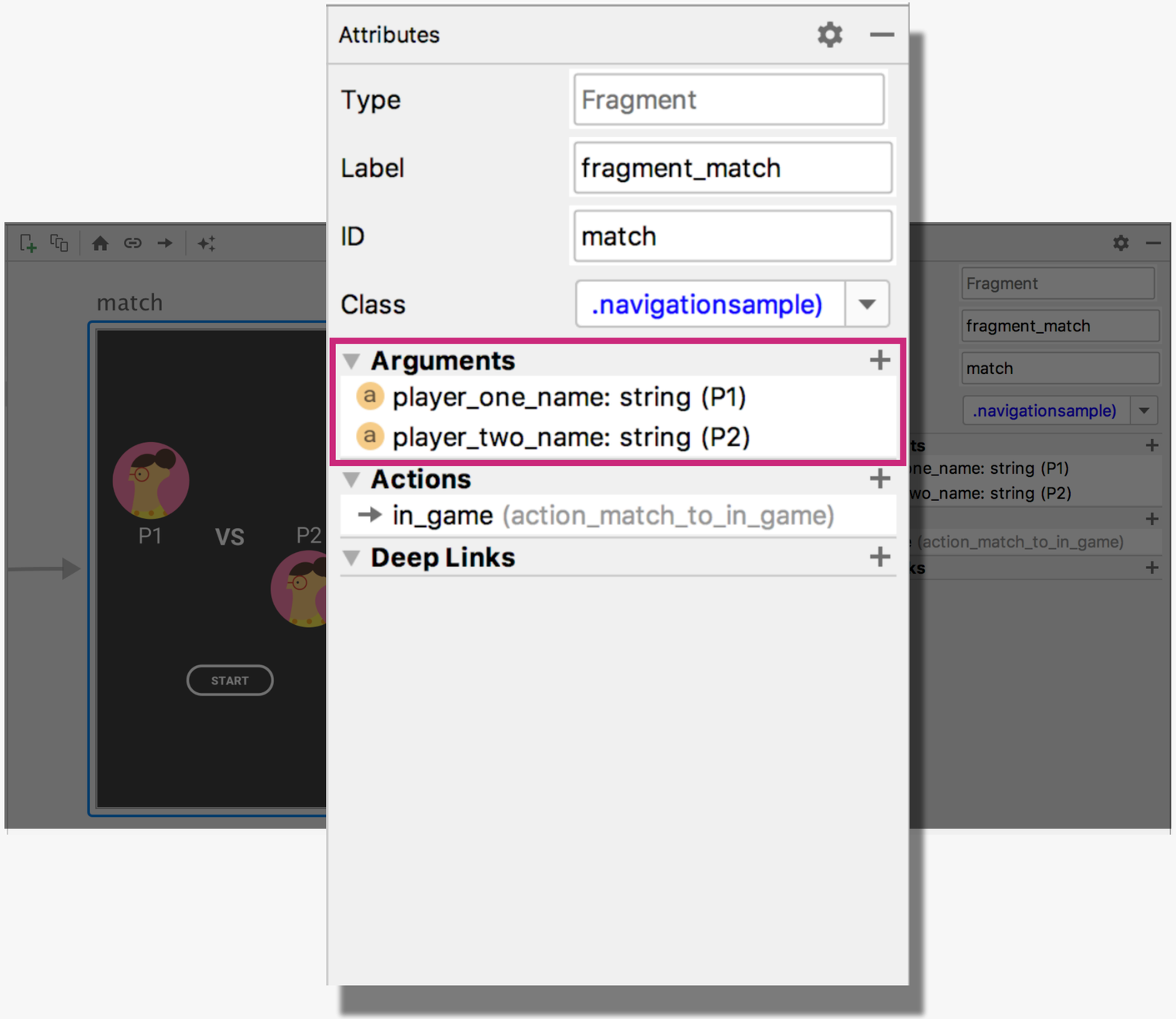
Destinations

- Arguments
- Deep Links
- Actions



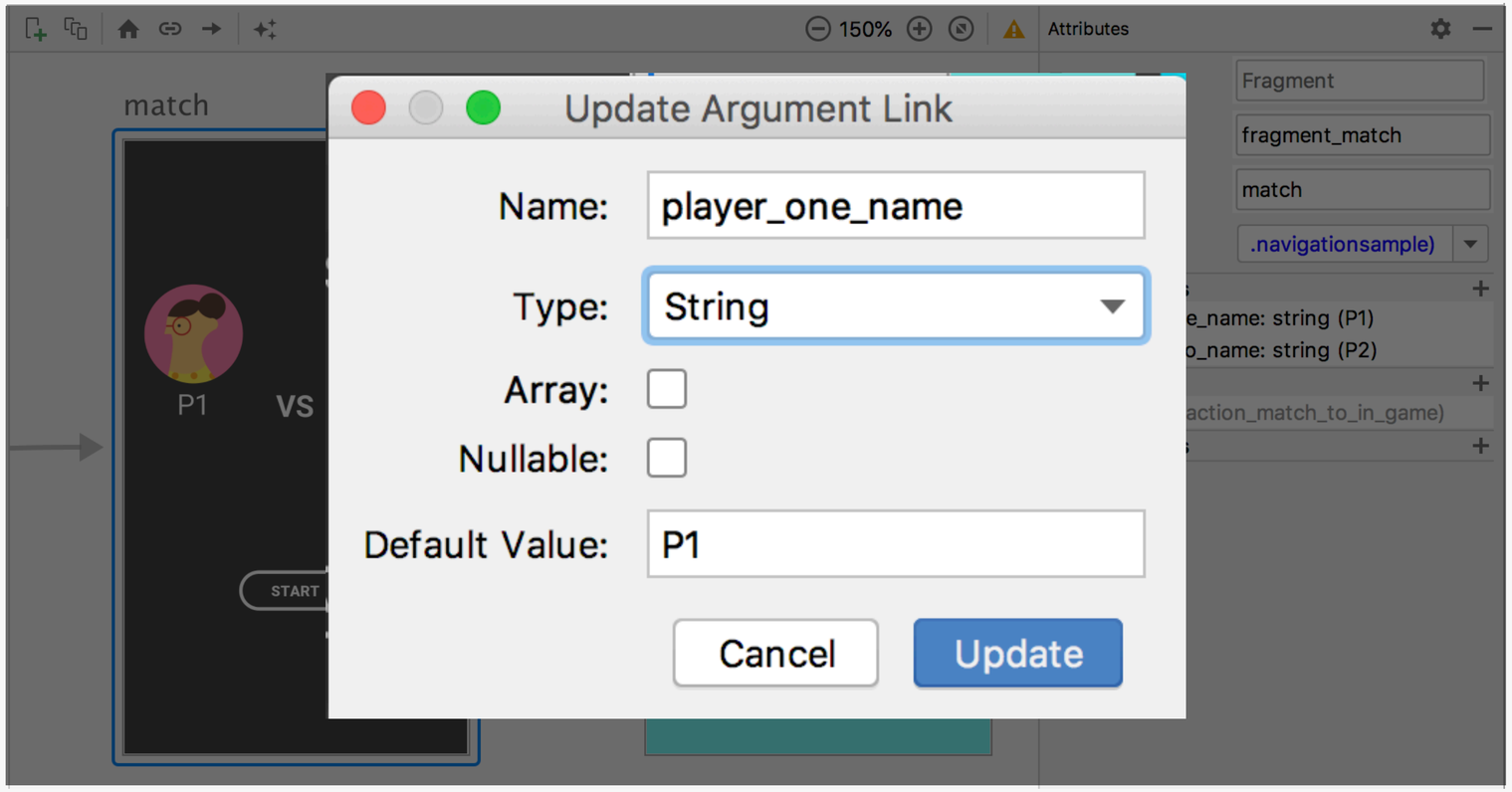
Destinations

- **Arguments**
- Deep Links
- Actions



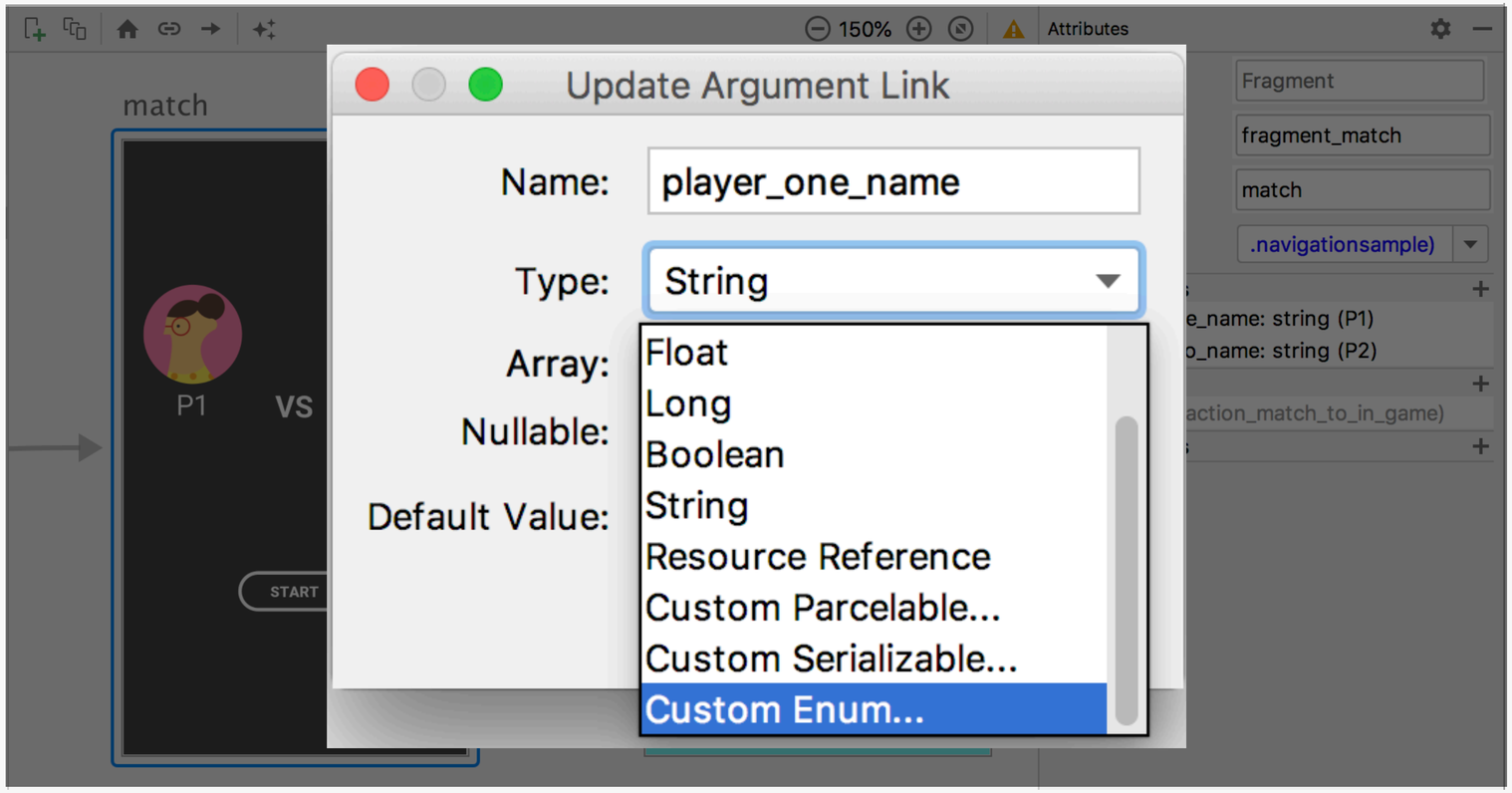
Destinations

- **Arguments**
- Deep Links
- Actions



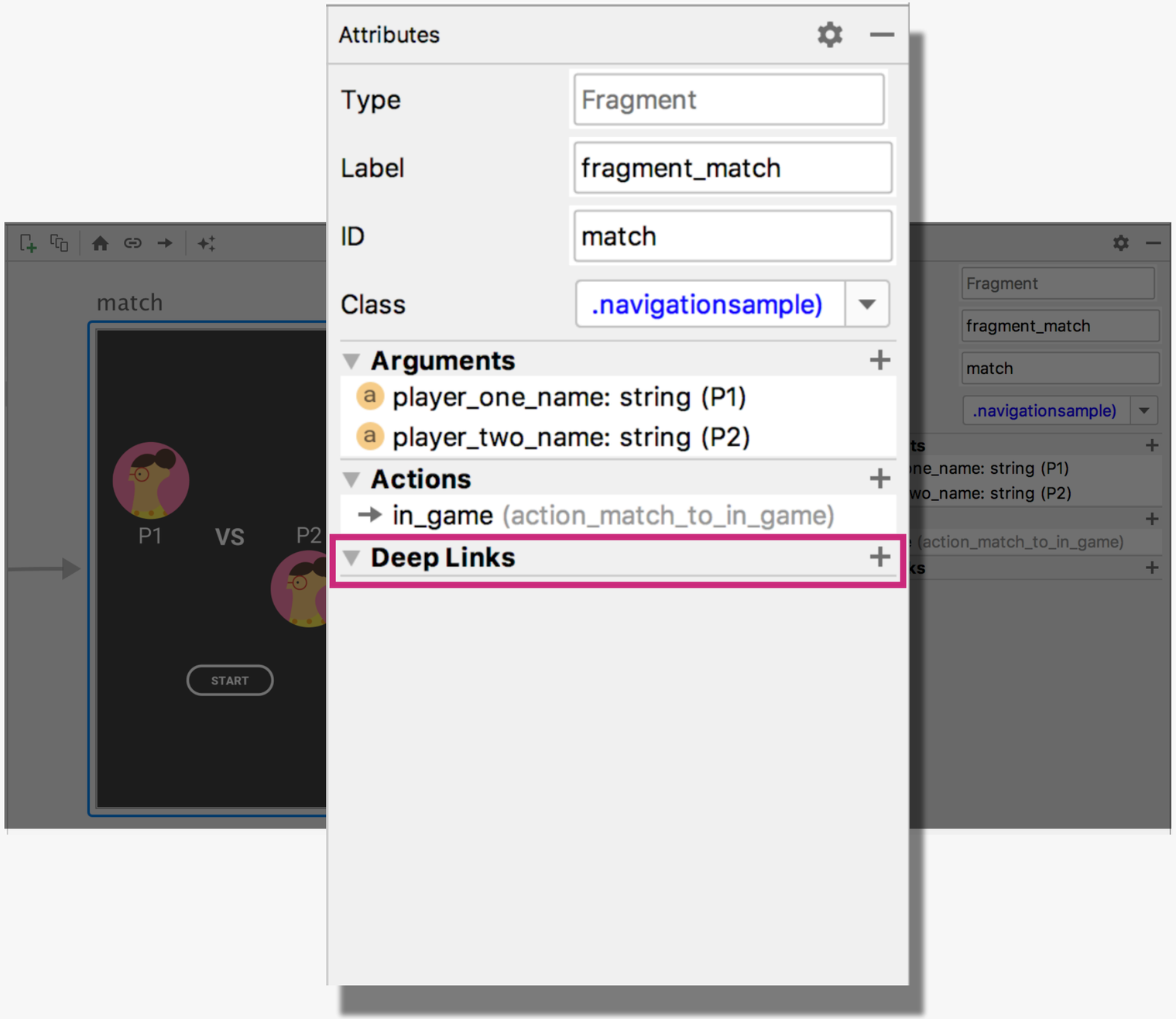
Destinations

- **Arguments**
- Deep Links
- Actions



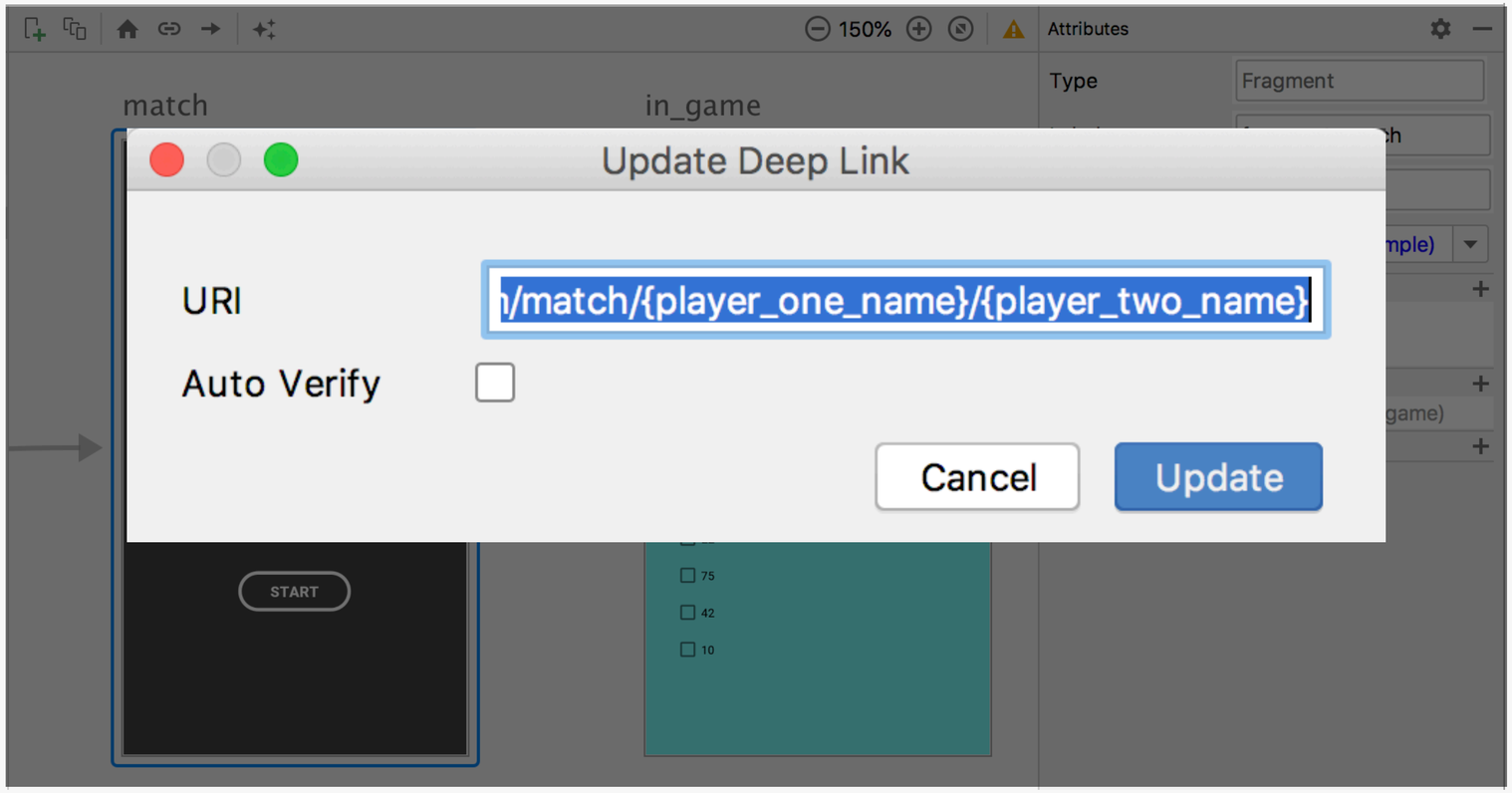
Destinations

- **Arguments**
- Deep Links
- Actions



Destinations

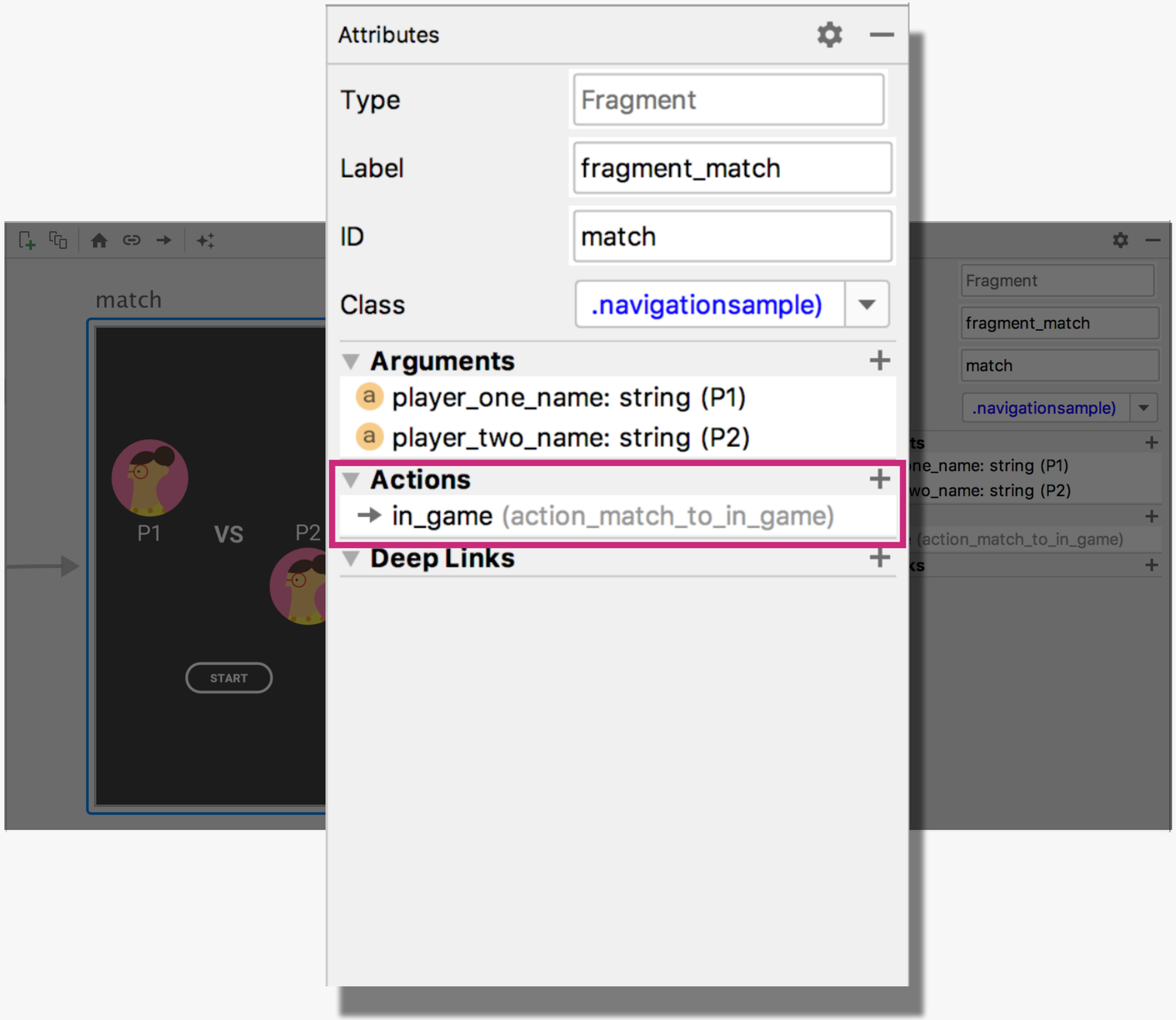
- Arguments
- **Deep Links**
- Actions




```
<deepLink app:uri="www.example.com/match/{player_one_name}/{player_two_name} " />
```

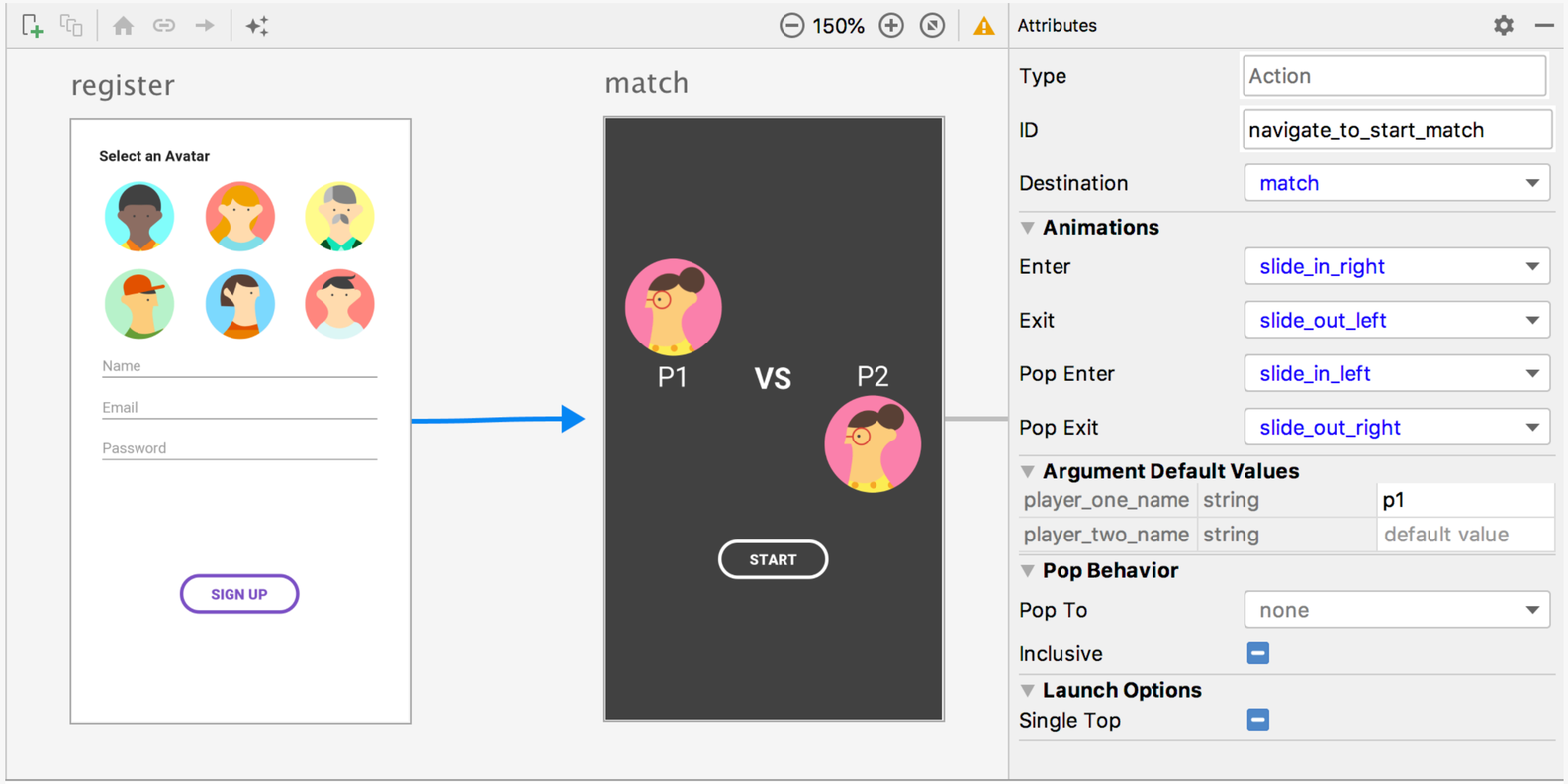
Destinations

- Arguments
- Deep Links
- **Actions**



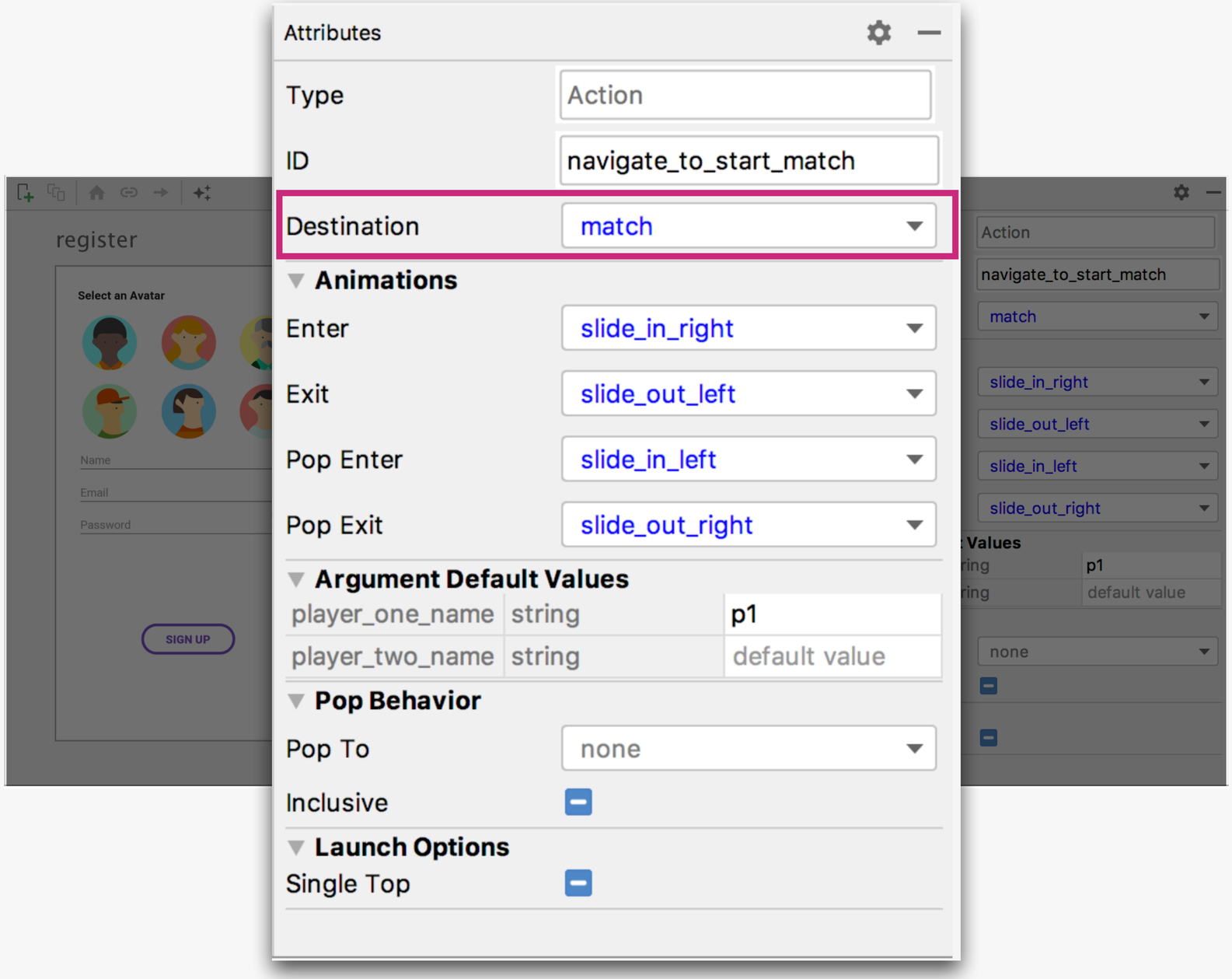
Actions

- **Destination**
- Argument Default Values
- Pop Behavior
- Launch Options
- Animations



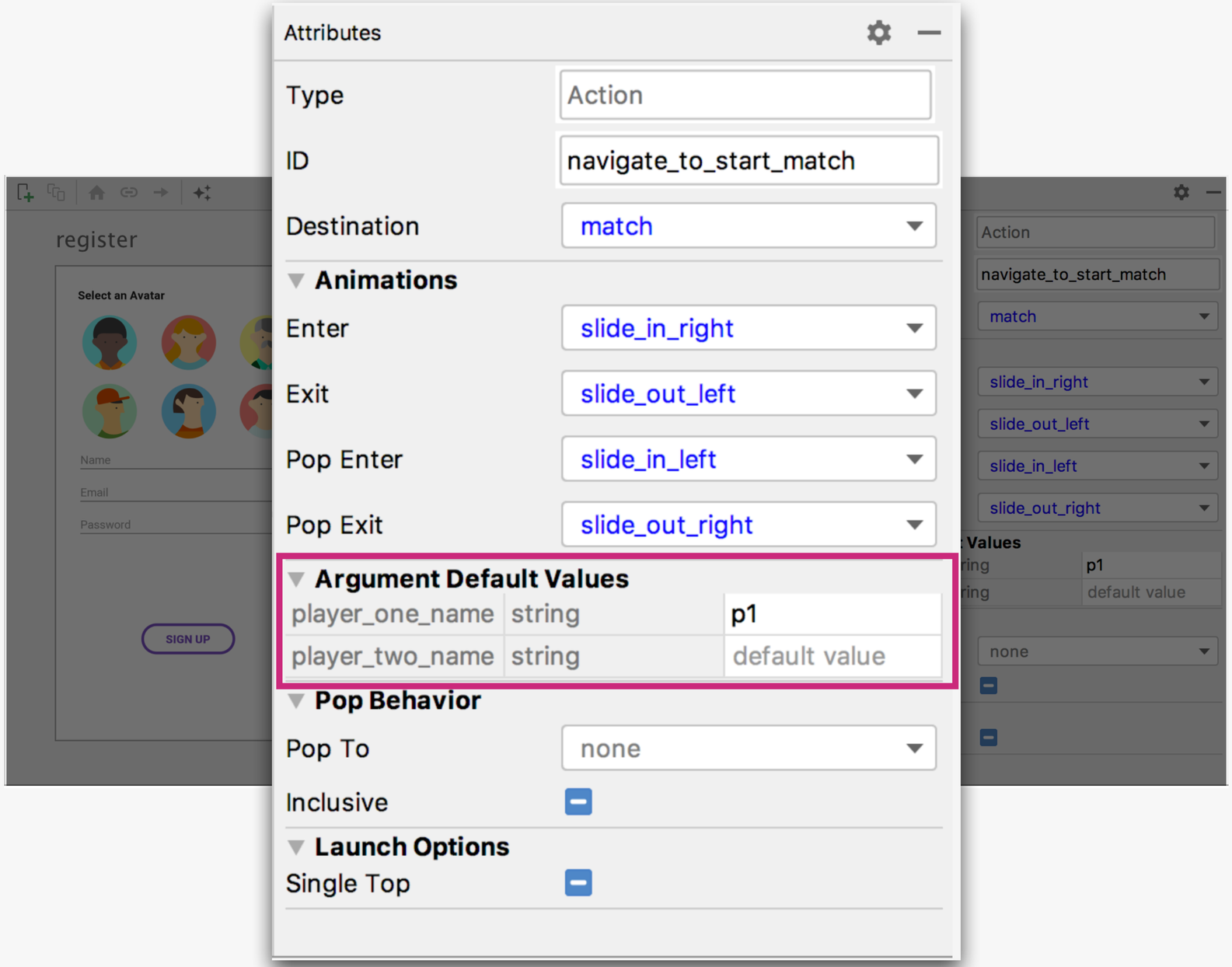
Actions

- **Destination**
- Argument Default Values
- Pop Behavior
- Launch Options
- Animations



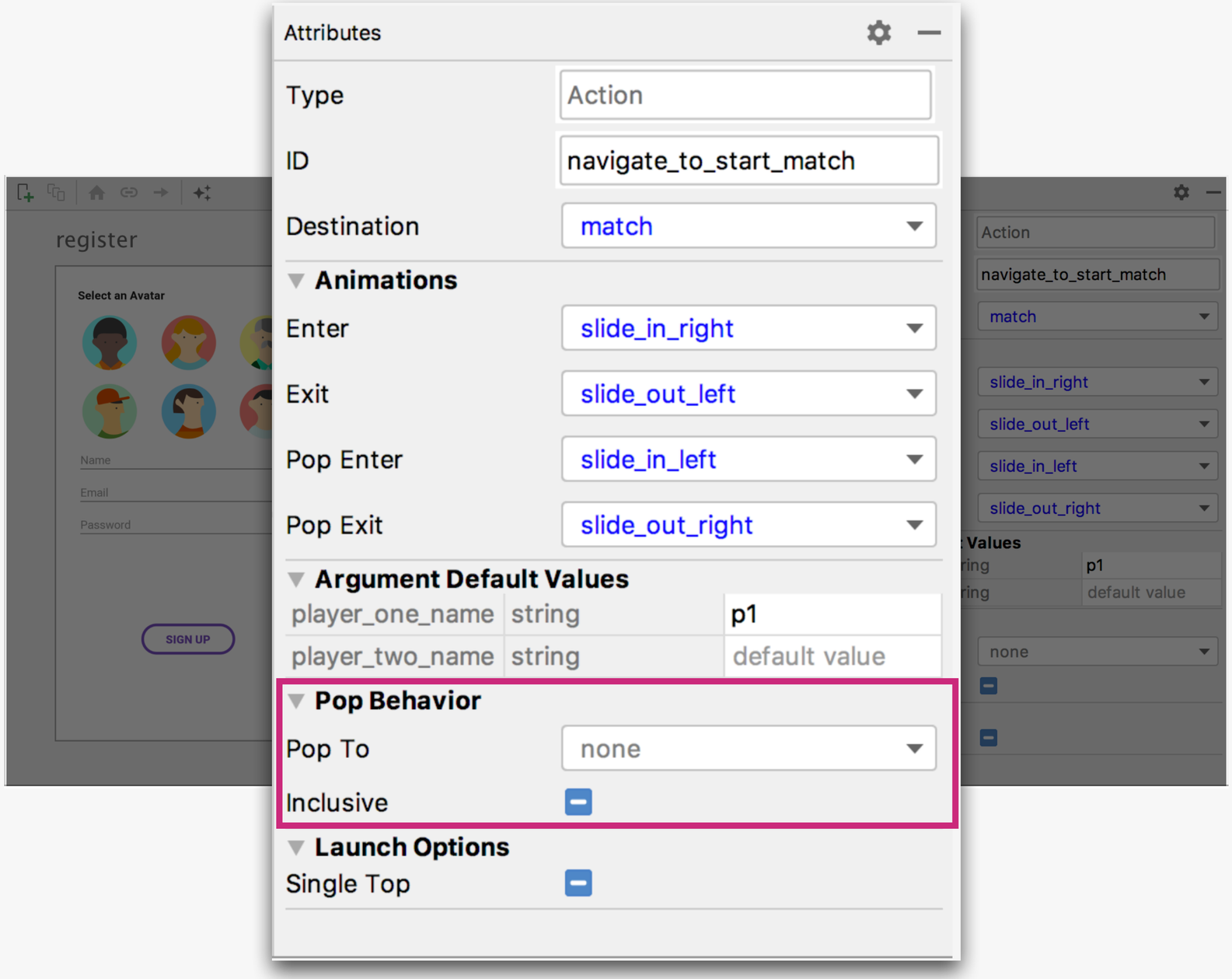
Actions

- Destination
- **Argument Default Values**
- Pop Behavior
- Launch Options
- Animations

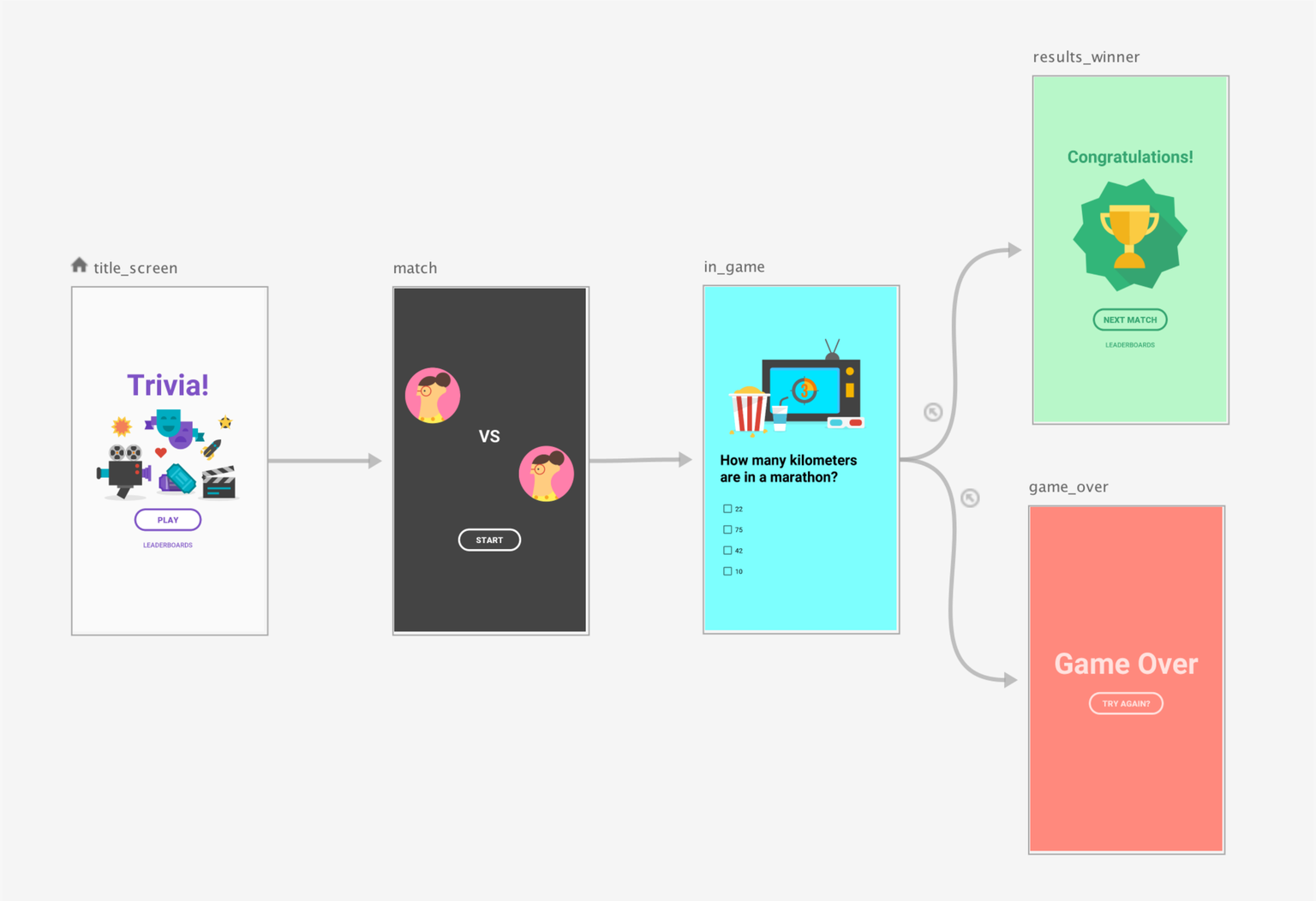


Actions

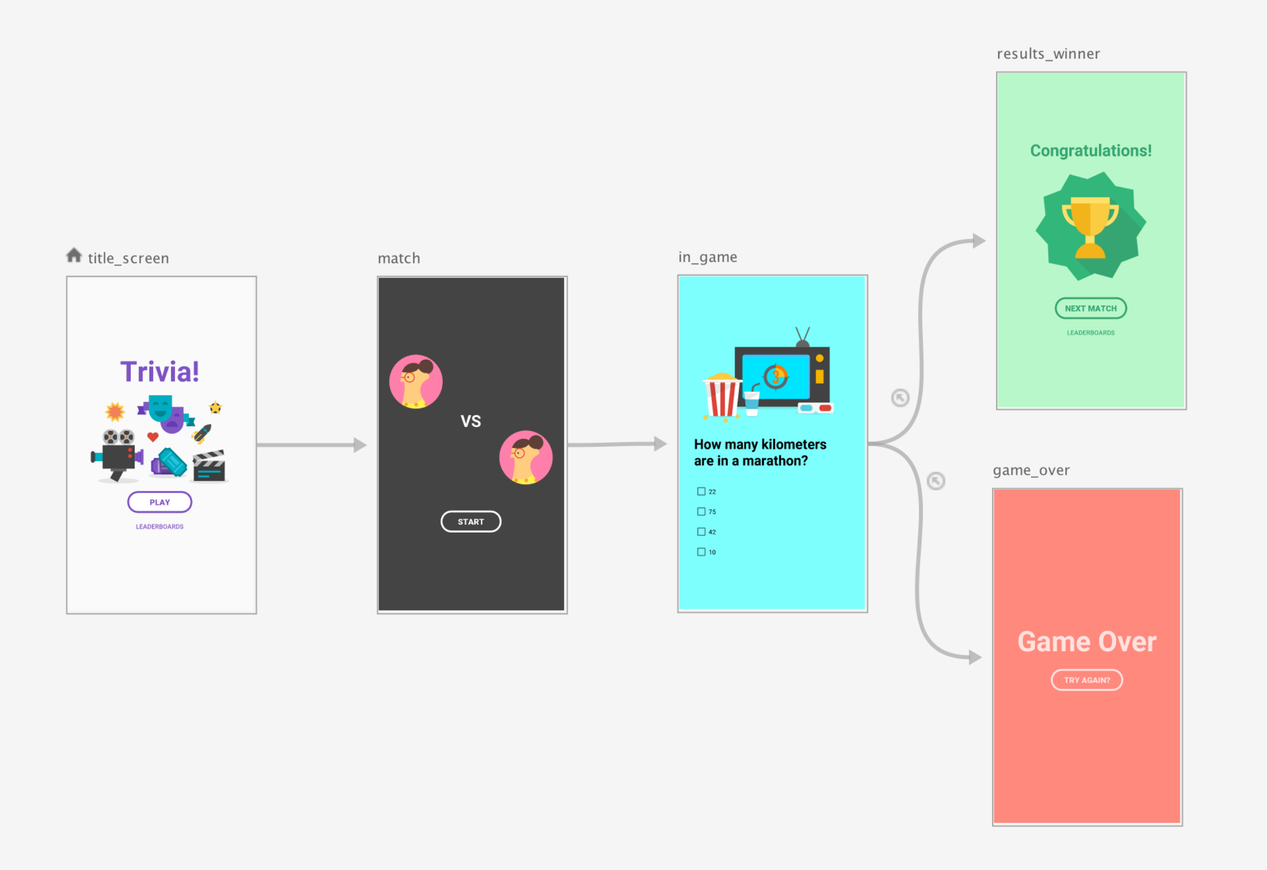
- Destination
- Argument Default Values
- **Pop Behavior**
- **Launch Options**
- Animations



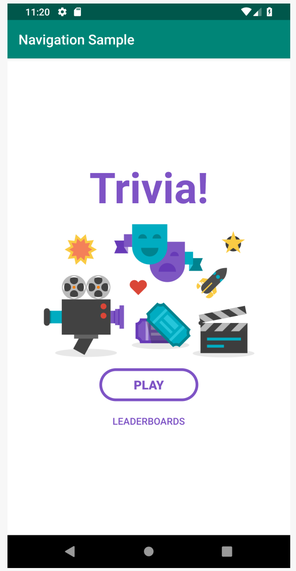
Stack Based Navigation



Stack Based Navigation

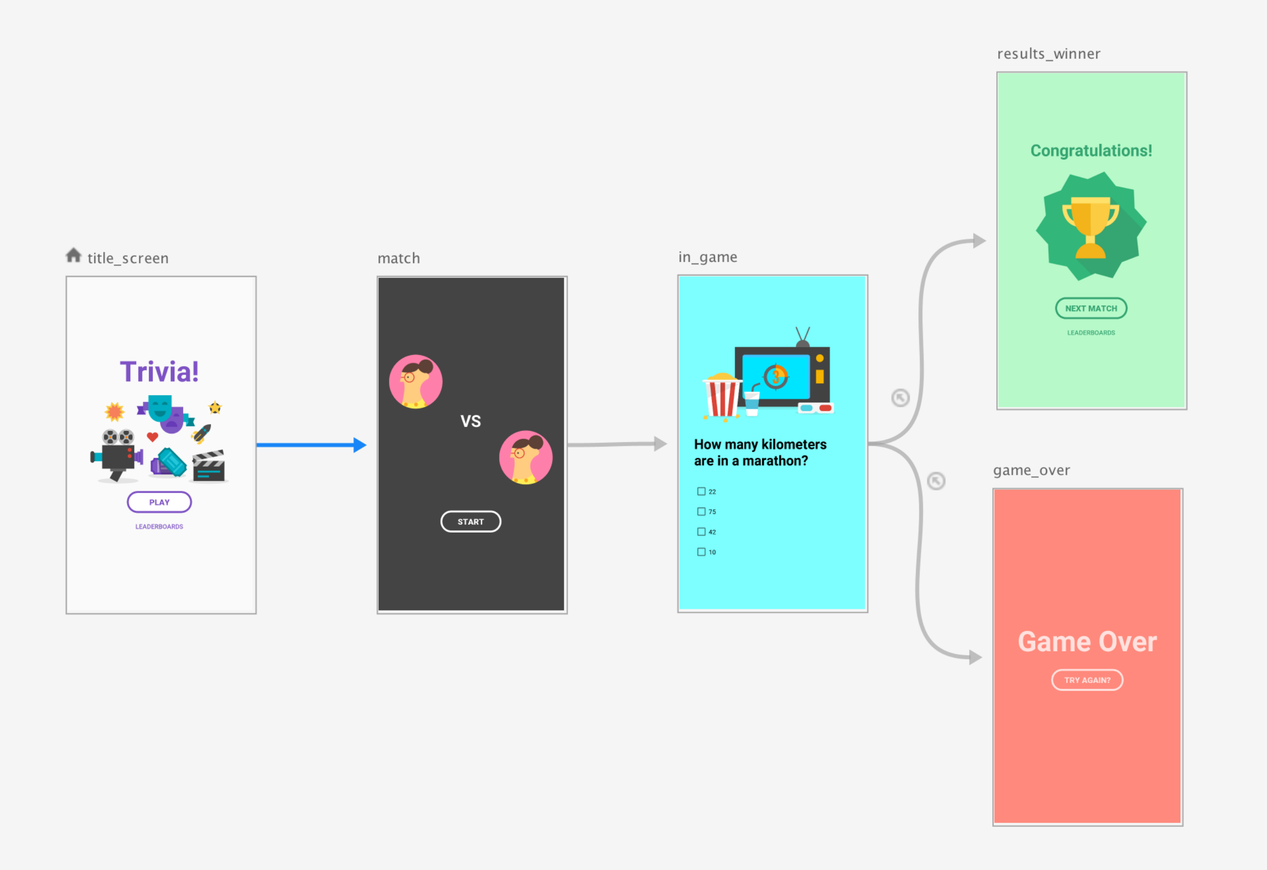


Navigation Graph

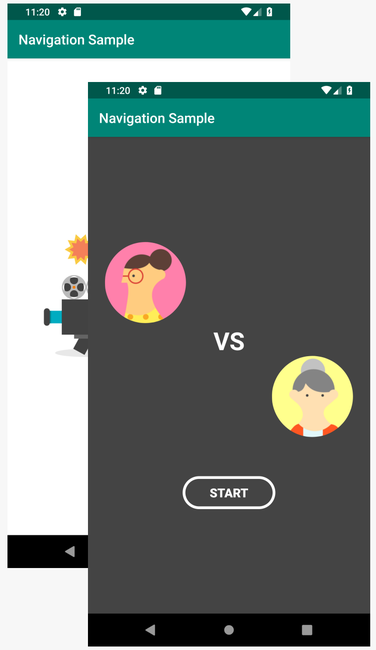


Stack

Stack Based Navigation

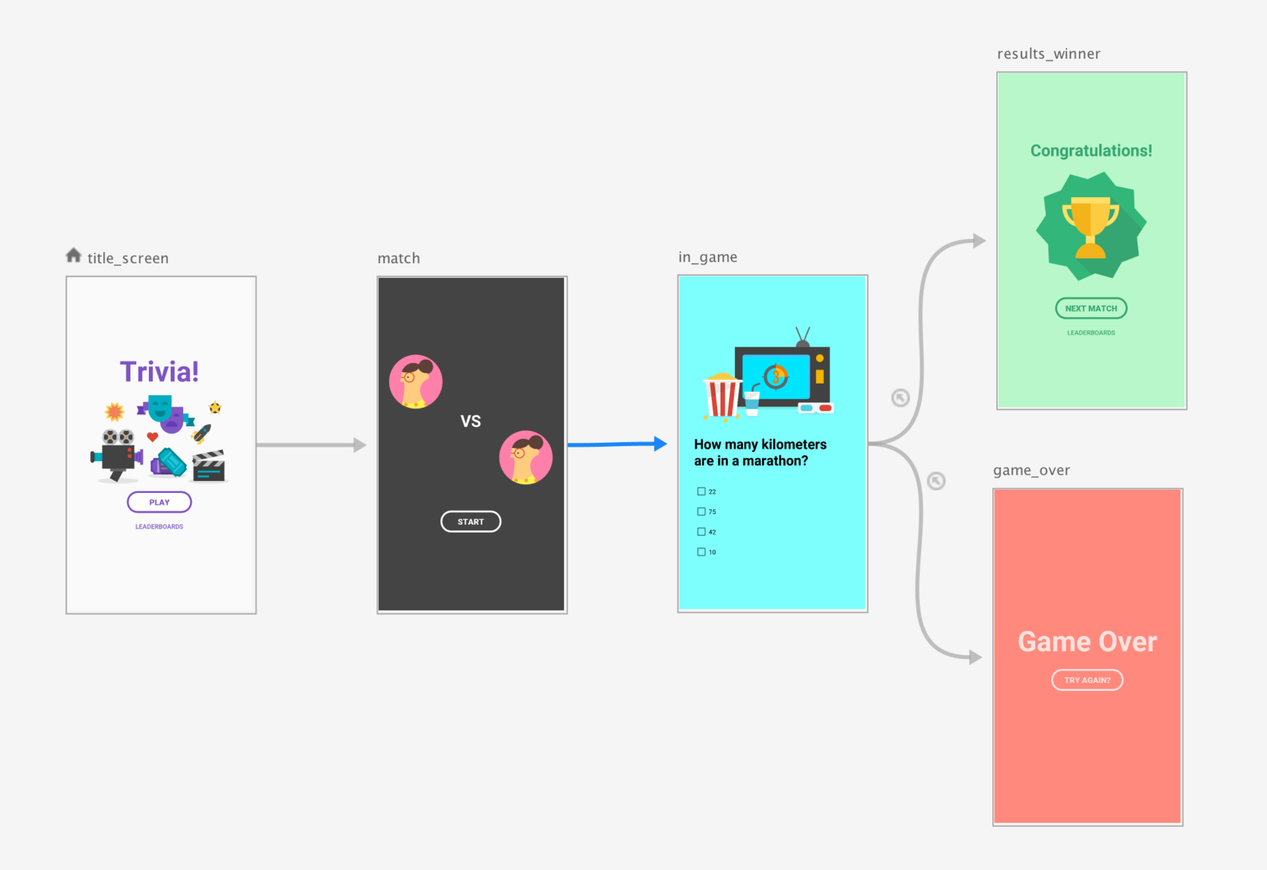


Navigation Graph

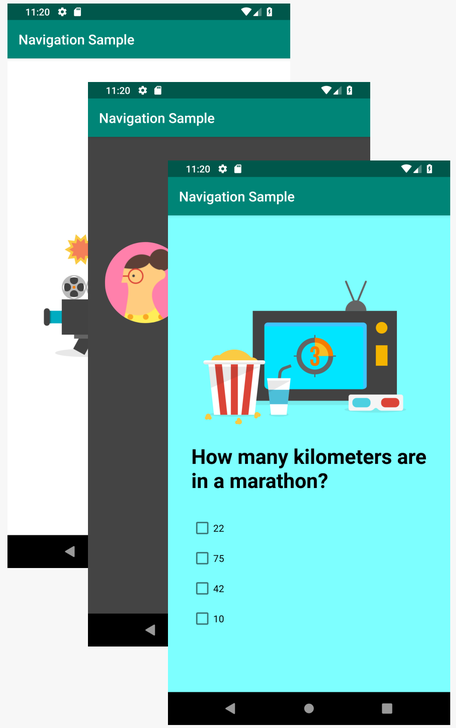


Stack

Stack Based Navigation

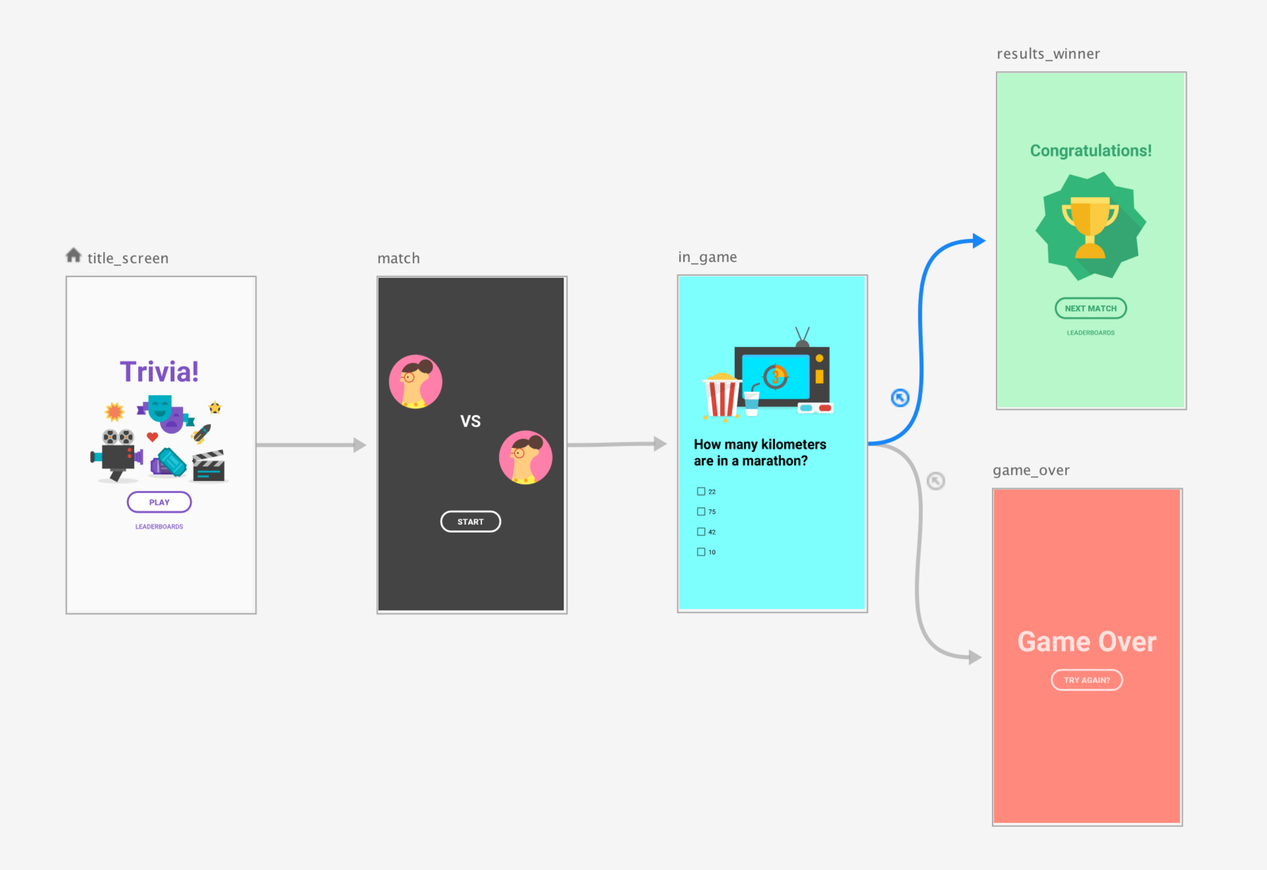


Navigation Graph

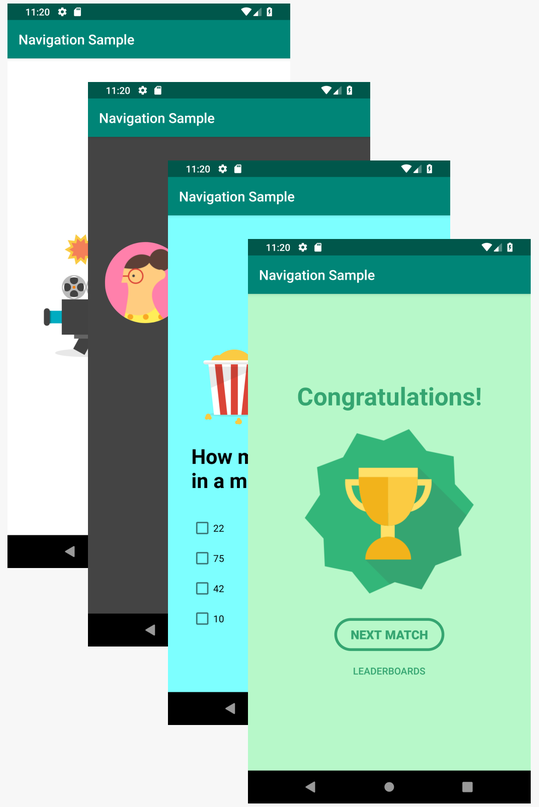


Stack

Stack Based Navigation



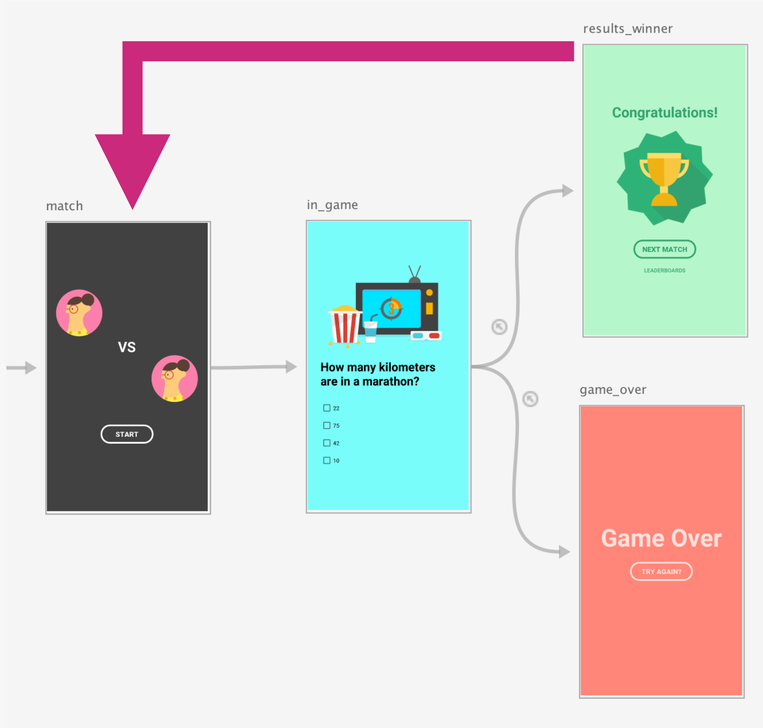
Navigation Graph



Stack

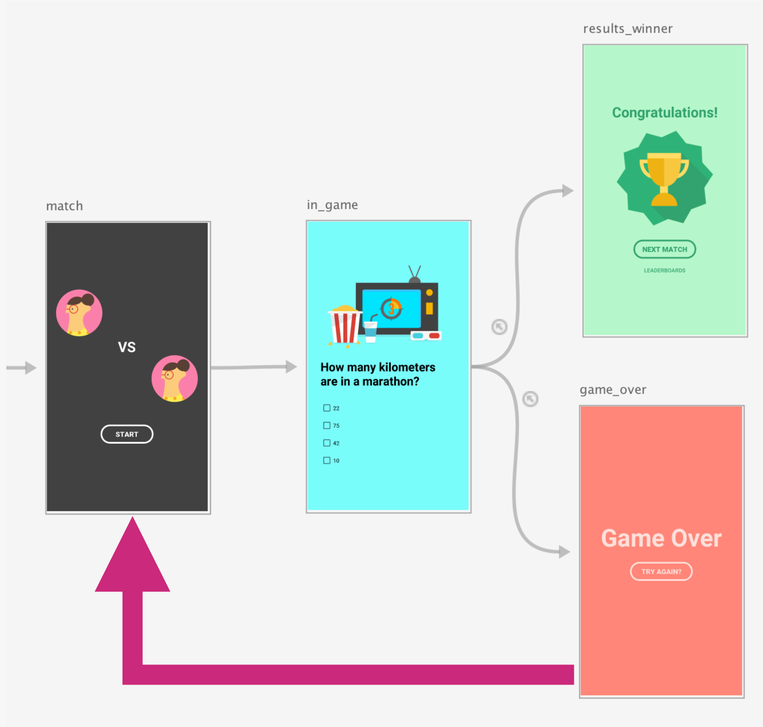
Popping Back to Match

```
<fragment android:id="@+id/match" ...>

<fragment android:id=“@+id/in_game" ...>

    <action
        android:id="@+id/action_in_game_to_results_winner"
        app:destination="@id/results_winner"
        app:popUpTo="@id/match" />
    <action
        android:id="@+id/action_in_game_to_game_over"
        app:destination="@id/game_over"
        app:popUpTo="@id/match" />
</fragment>

<fragment
        android:id="@+id/results_winner"
        android:name="com.example.android.navigationsample.ResultsWinner"/>

<fragment
        android:id="@+id/game_over"
        android:name=“com.example.android.navigationsample.GameOver" />
```



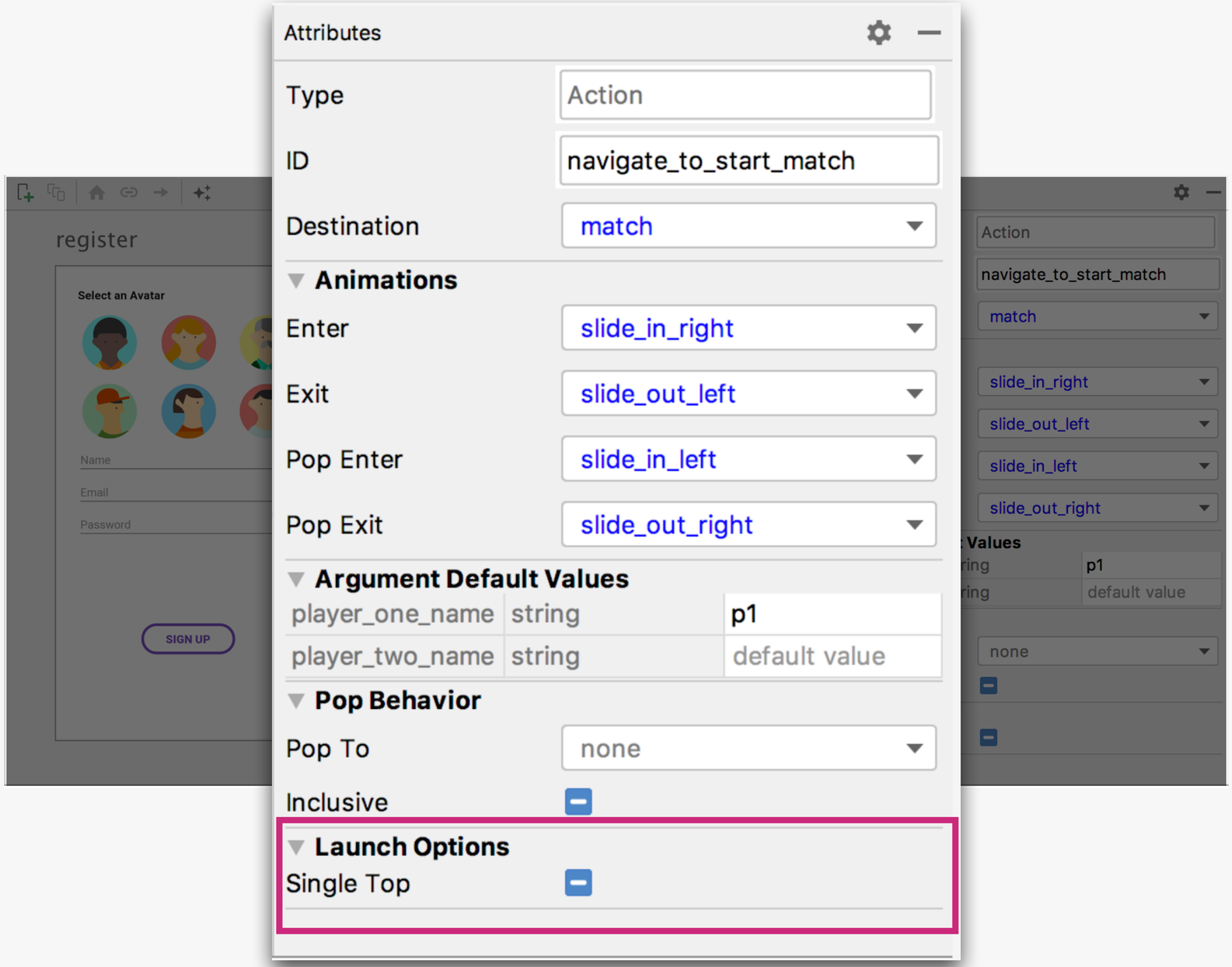
With Inclusive

```
<fragment android:id="@+id/match" ...>
<fragment android:id="@+id/in_game" ...>

    <action
        android:id="@+id/action_in_game_to_results_winner"
        app:destination="@id/results_winner"
        app:popUpTo="@id/match" />
    <action
        android:id="@+id/action_in_game_to_game_over"
        app:destination="@id/game_over"
        app:popUpTo="@id/in_game"
        app:popUpToInclusive="true"
 />
</fragment>

<fragment
        android:id="@+id/results_winner"
        android:name="com.example.android.navigationsample.ResultsWinner"/>

<fragment
        android:id="@+id/game_over"
        android:name="com.example.android.navigationsample.GameOver" />
```



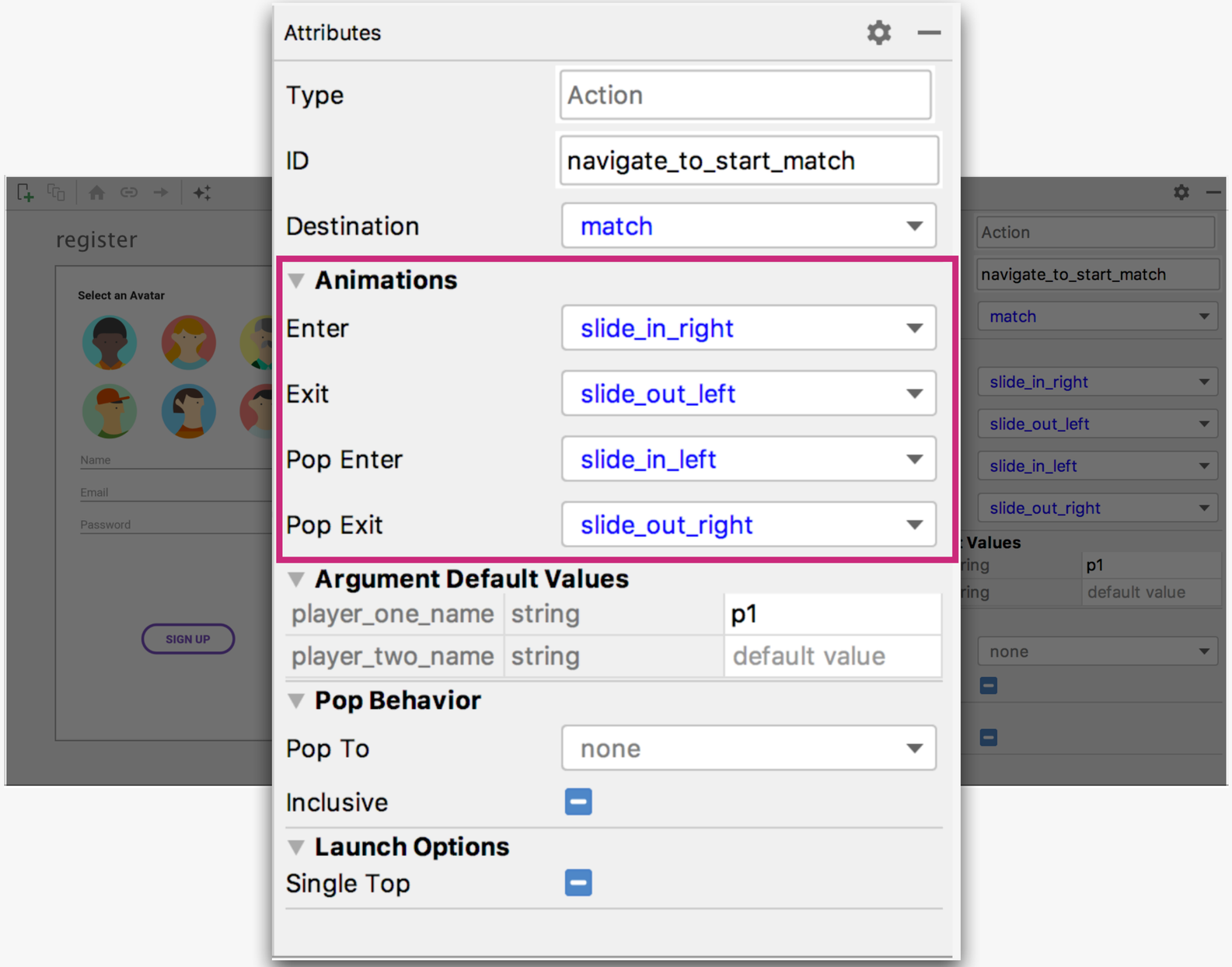
Actions

- Destination
- Argument Default Values
- Pop Behavior
- **Launch Options**
- Animations

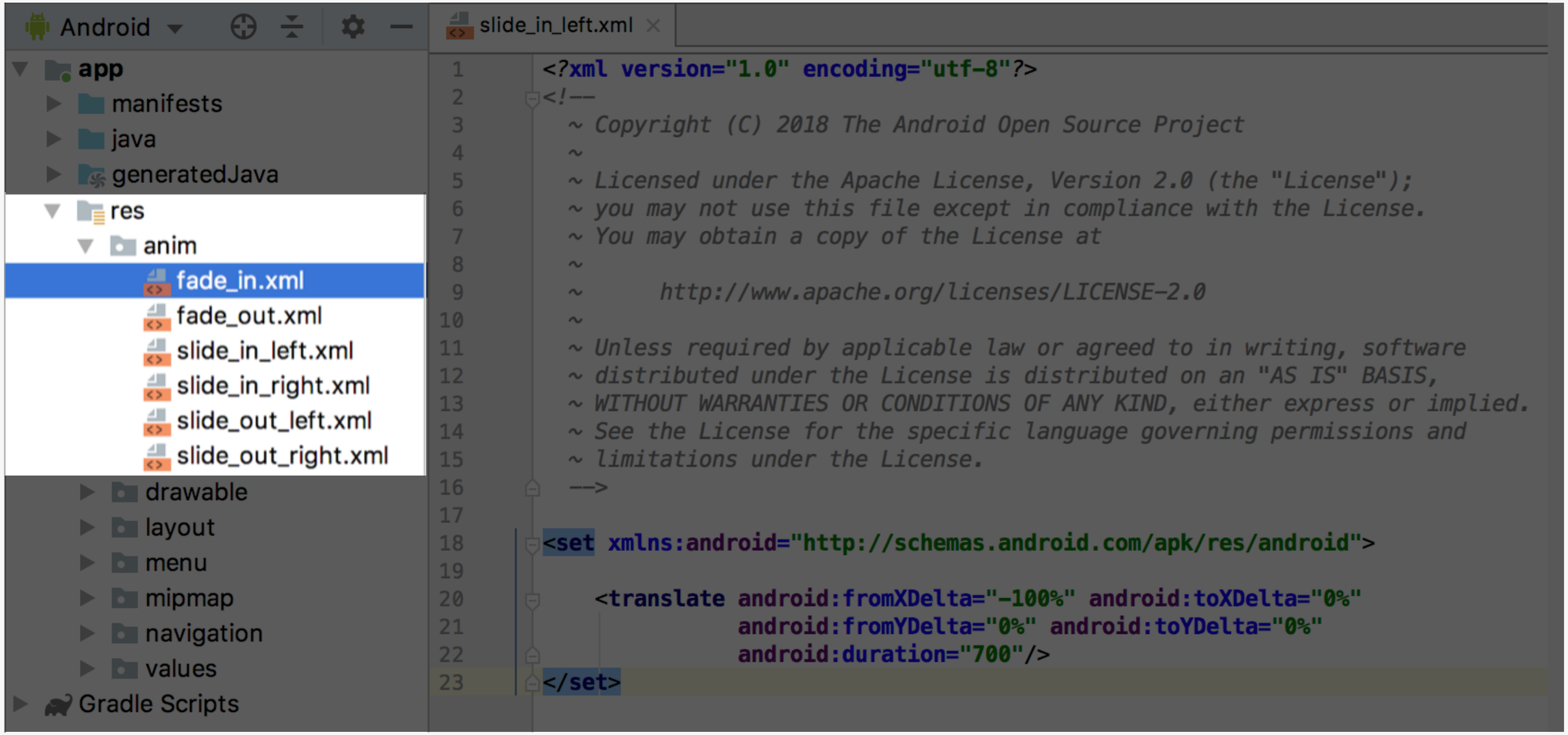


Actions

- Destination
- Argument Default Values
- Pop Behavior
- Launch Options
- **Animations**



CUSTOM ANIMATIONS



Inside Activity

```
<FrameLayout 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">

    <fragment
        android:id=“@+id/nav_host_fragment”
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:name="androidx.navigation.fragment.NavHostFragment"
        app:defaultNavHost="true"
        app:navGraph=“@navigation/navigation_graph”/>

</FrameLayout>
```

Include NavHostFragment

```
<FrameLayout 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">

    <fragment
        android:id="@+id/nav_host_fragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:name="androidx.navigation.fragment.NavHostFragment"
        app:defaultNavHost="true"
        app:navGraph="@navigation/navigation_graph" />

</FrameLayout>
```

HOSTING NAVIGATION

Navigation Graph

NavHostFragment

NavHostFragment Initialization

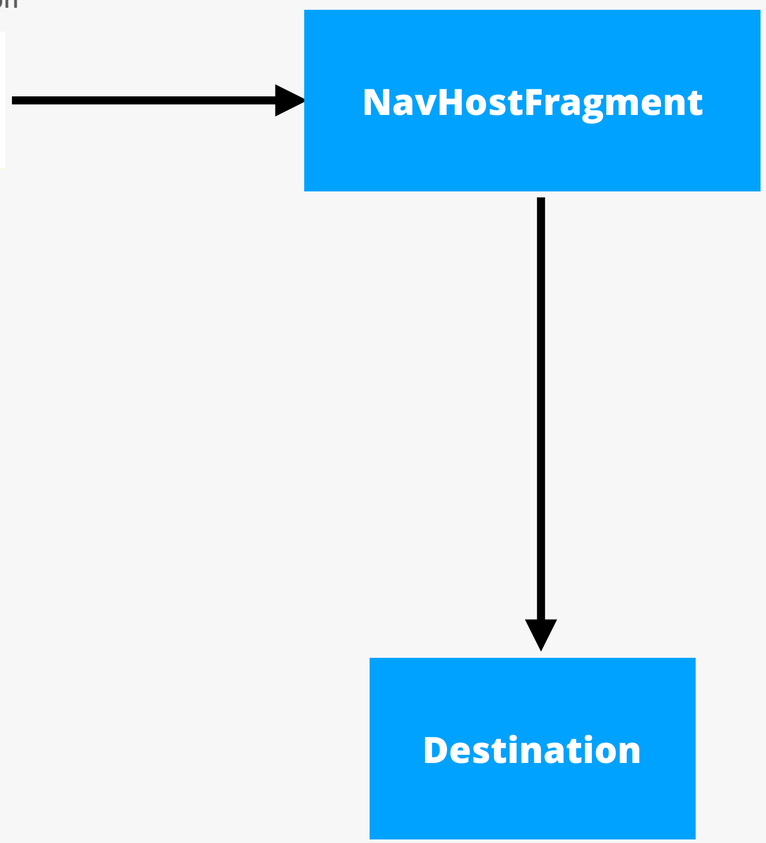
1. **Create NavController**

   ...

2. Navigate to first Destination

HOSTING NAVIGATION

Navigation Graph



NavHostFragment Initialization

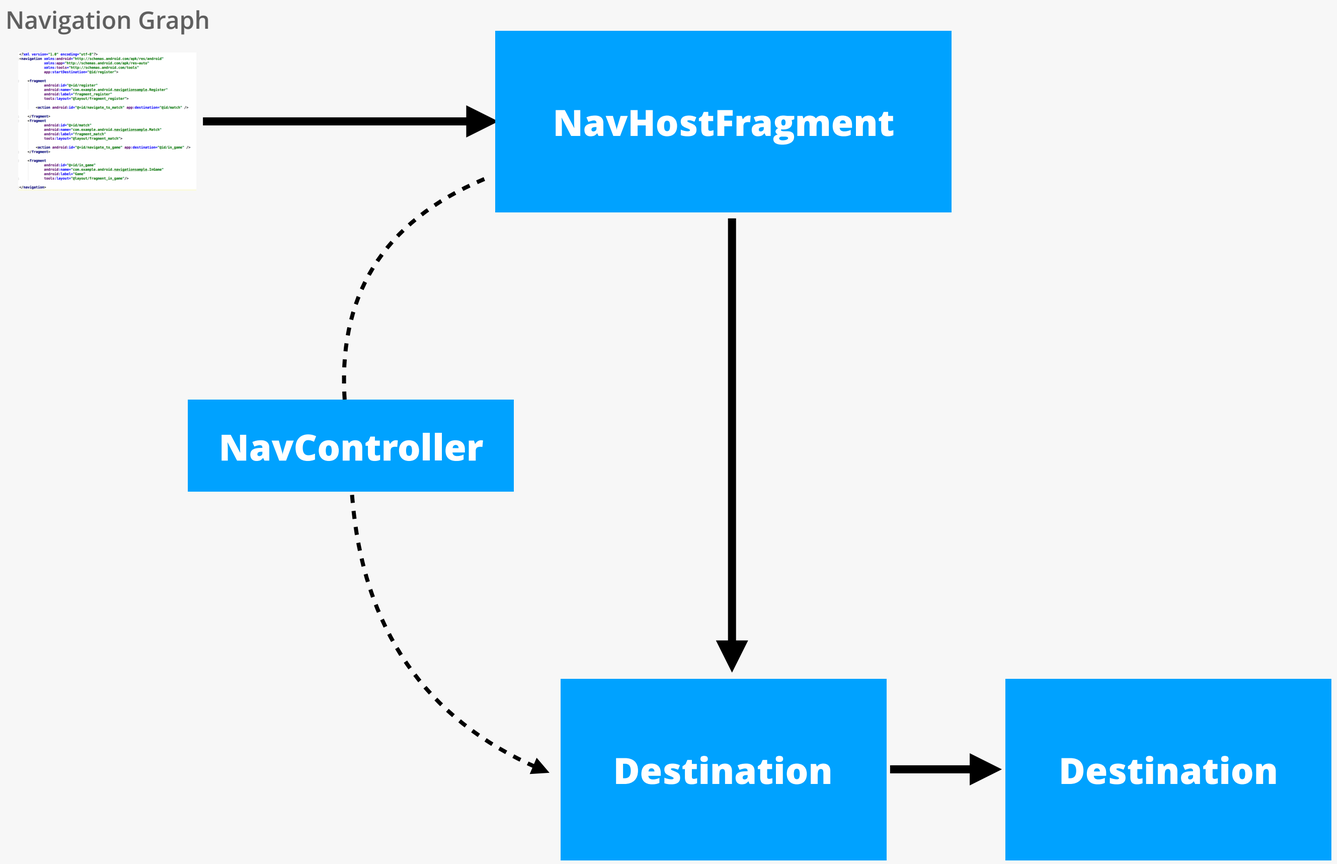
1. Create NavController

   ...

2. Navigate to first Destination

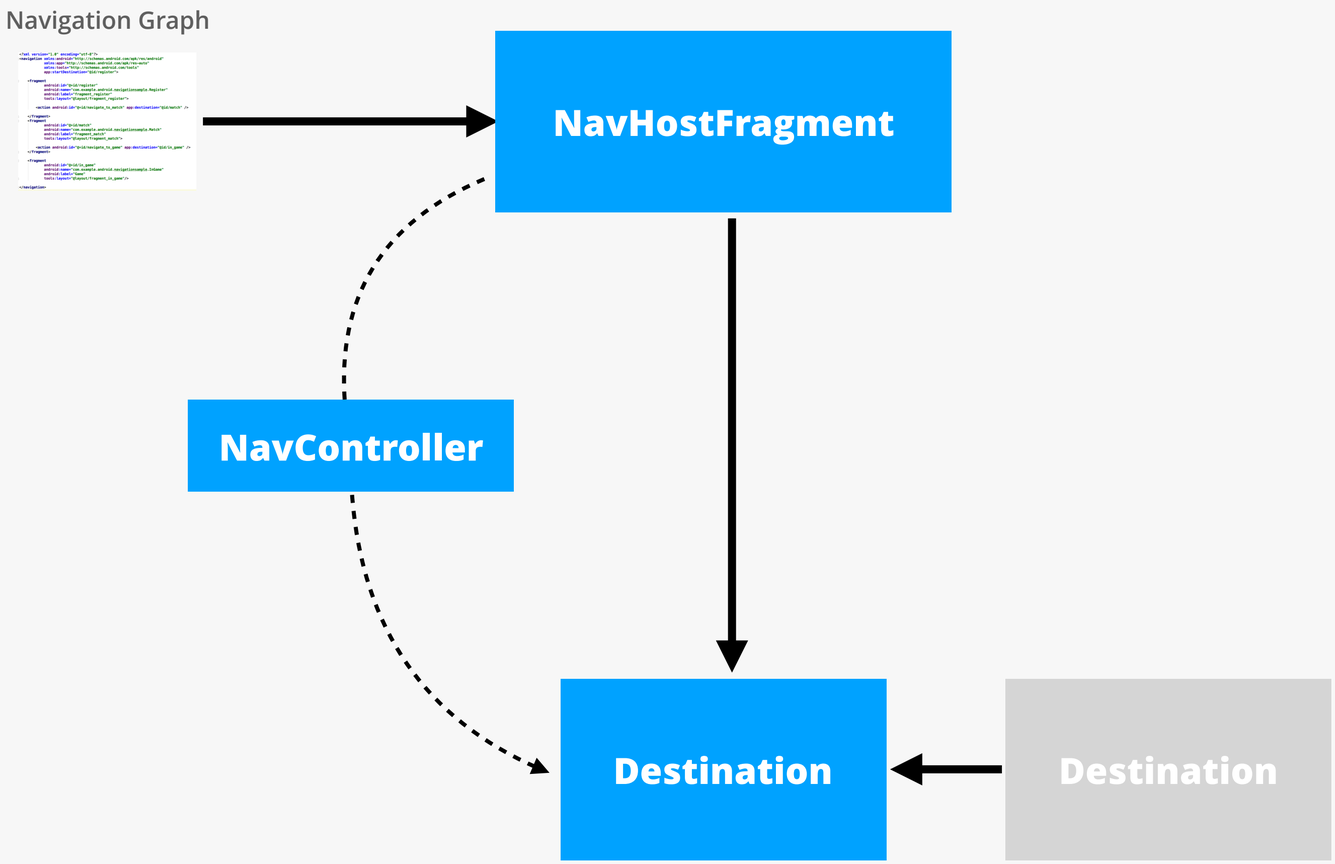
```
<navigation xmlns:android="..."
    xmlns:app="..."
    xmlns:tools=..."
    android:id="@+id/contacts_graph"
app:startDestination="@id/contacts_list">

    <fragment
        android:id="@+id/contacts_list"
        android:name=".ContactsListFragment"
        android:label="Contacts"
        tools:layout="@layout/fragment_contacts_list">
```

HOSTING NAVIGATION




```
navController.navigate(...)
```

HOSTING NAVIGATION




```
navController.popBackStack()
```

Accessing NavController

Fragment

```
class MyDestinationFragment : Fragment()

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        val controller: NavController = findNavController()
    }
}
```

Activity

```
class MainActivity : AppCompatActivity() {

    private lateinit var navController: NavController

     override fun onCreate(savedInstanceState: Bundle?) {
         super.onCreate(savedInstanceState)
         setContentView(R.layout.activity_main)

         val controller: NavController = findNavController(R.id.nav_host_fragment)
}
```

Navigate to Another Destination

```
class MyDestinationFragment : Fragment()

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        val controller: NavController = findNavController()

        view.findViewById<Button>(R.id.next)).setOnClickListener {

            controller.navigate(R.id.action_id)

        }

    }
}
```

Pop Back to Previous Destination

```
class MyDestinationFragment : Fragment()

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        val controller: NavController = findNavController()

        view.findViewById<Button>(R.id.next)).setOnClickListener {

            controller.navigate(R.id.action_id)

        }

        view.findViewById<Button>(R.id.previous)).setOnClickListener {

            controller.popBackStack()


        }

    }
}
```

Operations

Navigate to Actions or Destinations by Resource ID

```
fun navigate(@IdRes resId: Int)
fun navigate(@IdRes resId: Int, args: Bundle?)
fun navigate(@IdRes resId: Int, args: Bundle?, navOptions: NavOptions?)
fun navigate(@IdRes resId: Int, args: Bundle?, navOptions: NavOptions?, navigatorExtras: Navigator.Extras?)
```

Navigate using Safe Args generated classes

```
fun navigate(directions: NavDirections)
fun navigate(directions: NavDirections, navOptions: NavOptions?)
fun navigate(directions: NavDirections, navigatorExtras: Navigator.Extras)
```

Back Navigation

```
fun popBackStack(): Boolean
fun popBackStack(@IdRes destinationId: Int, inclusive: Boolean): Boolean
```

Operations

Navigate to Actions or Destinations by Resource ID

Via Destination or Action ID with options

```
fun navigate(@IdRes resId: Int)
fun navigate(@IdRes resId: Int, args: Bundle?)
fun navigate(@IdRes resId: Int, args: Bundle?, navOptions: NavOptions?)
fun navigate(@IdRes resId: Int, args: Bundle?, navOptions: NavOptions?, navigatorExtras: Navigator.Extras?)
```

Navigate using Safe Args generated classes

```
fun navigate(directions: NavDirections)
fun navigate(directions: NavDirections, navOptions: NavOptions?)
fun navigate(directions: NavDirections, navigatorExtras: Navigator.Extras)
```

Back Navigation

```
fun popBackStack(): Boolean
fun popBackStack(@IdRes destinationId: Int, inclusive: Boolean): Boolean
```

Operations

Navigate to Actions or Destinations by Resource ID

```
fun navigate(@IdRes resId: Int)
fun navigate(@IdRes resId: Int, args: Bundle?)
fun navigate(@IdRes resId: Int, args: Bundle?, navOptions: NavOptions?)
fun navigate(@IdRes resId: Int, args: Bundle?, navOptions: NavOptions?, navigatorExtras: Navigator.Extras?)
```

Navigate using Safe Args generated classes

```
fun navigate(directions: NavDirections)
fun navigate(directions: NavDirections, navOptions: NavOptions?)
fun navigate(directions: NavDirections, navigatorExtras: Navigator.Extras)
```

Back Navigation

```
fun popBackStack(): Boolean
fun popBackStack(@IdRes destinationId: Int, inclusive: Boolean): Boolean
```

Operations

Navigate to Actions or Destinations by Resource ID

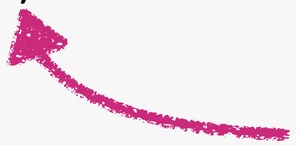
```
fun navigate(@IdRes resId: Int)
fun navigate(@IdRes resId: Int, args: Bundle?)
fun navigate(@IdRes resId: Int, args: Bundle?, navOptions: NavOptions?)
fun navigate(@IdRes resId: Int, args: Bundle?, navOptions: NavOptions?, navigatorExtras: Navigator.Extras?)
```

Navigate using Safe Args generated classes

```
fun navigate(directions: NavDirections)
fun navigate(directions: NavDirections, navOptions: NavOptions?)
fun navigate(directions: NavDirections, navigatorExtras: Navigator.Extras)
```

Back Navigation

```
fun popBackStack(): Boolean
fun popBackStack(@IdRes destinationId: Int, inclusive: Boolean): Boolean
```



Pop back to destination up the stack

Safe Args

Safe Args Dependencies

Safe Args

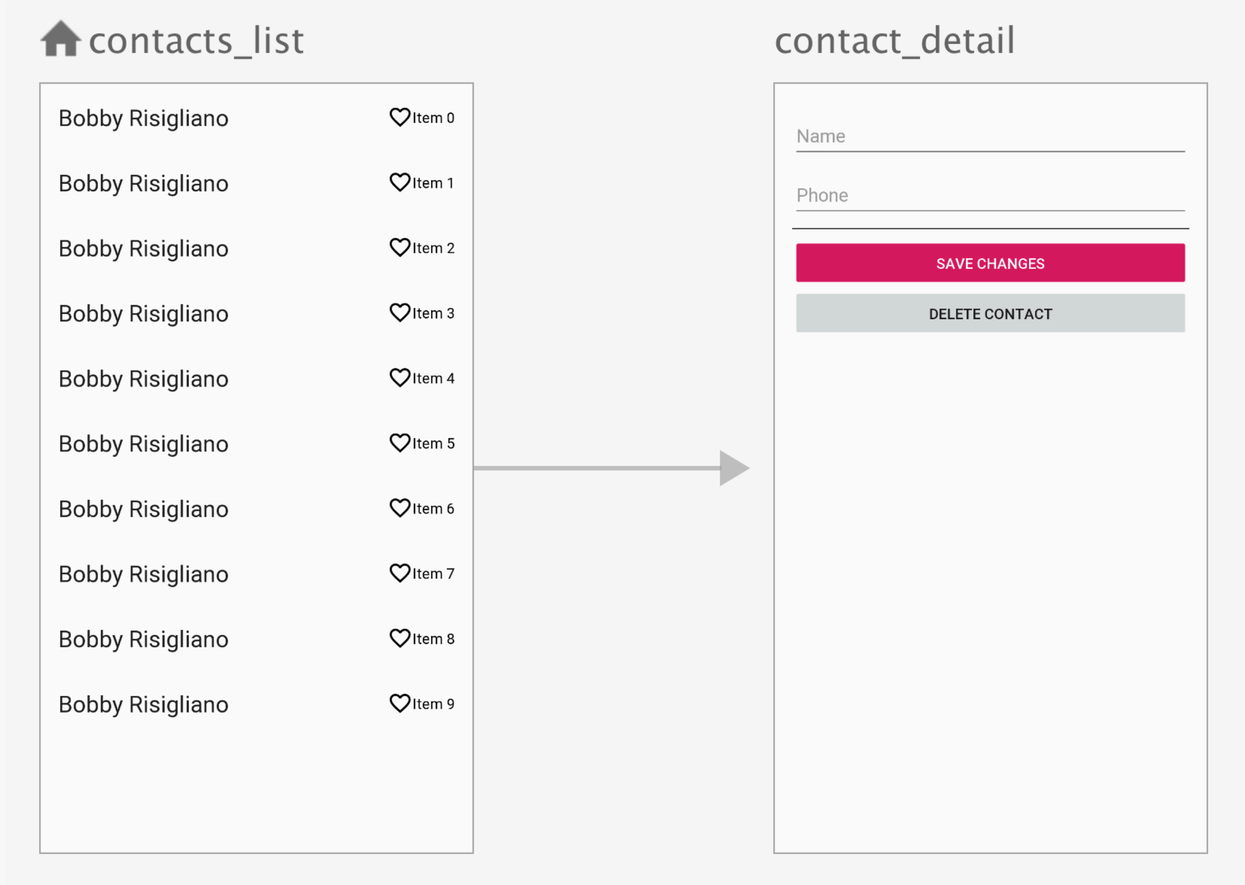
Project build.gradle:

```
dependencies {
    ...
    classpath “androidx.navigation:navigation-safe-args-gradle-plugin:$version"
}
```

Module build.gradle:

```
apply plugin: 'androidx.navigation.safeargs.kotlin'
```

Generated Directions + Arguments



Destination Requires

- `Contact Id : String`

Generated Arguments + Directions

```
<navigation
    android:id="@+id/contacts_graph"
    app:startDestination="@id/contacts_list">

    <fragment android:id="@+id/contacts_list" android:name=".ContactsListFragment">

        <action
            android:id="@+id/to_contact_detail"
            app:destination="@id/contact_detail" />

   </fragment>

    <fragment android:id=“@+id/contact_detail"
              android:name=".ContactDetailFragment">

        <argument android:name=“contactId”
                  app:argType=“string"
                  app:nullable="true"
                  app:defaultValue="@null" />

    </fragment>
</navigation>
```

Generated Arguments

```
// Create instance given a contactId : String
data class ContactDetailFragmentArgs(val contactId: String? = null) : NavArgs {

    // Output to a bundle
    fun toBundle(): Bundle { ... }

    companion object {

        // Create instance from a Bundle
        @JvmStatic
        fun fromBundle(bundle: Bundle): ContactDetailFragmentArgs { ... }
    }
}
```

Navigation Graph

```
<fragment android:id="@+id/contact_detail"
          android:name=".ContactDetailFragment">

    <argument android:name="contactId"
              app:argType="string"
              app:nullable="true"
              app:defaultValue="@null" />

</fragment>
```

Generated Arguments

```
// Create instance given a contactId : String
data class ContactDetailFragmentArgs(val contactId: String? = null) : NavArgs {

    // Output to a bundle
    fun toBundle(): Bundle { ... }

    companion object {

        // Create instance from a Bundle
        @JvmStatic
        fun fromBundle(bundle: Bundle): ContactDetailFragmentArgs { ... }
    }
}
```

Create from Bundle in fragment

Navigation Graph

```
<fragment android:id="@+id/contact_detail"
          android:name=".ContactDetailFragment">

    <argument android:name="contactId"
              app:argType="string"
              app:nullable="true"
              app:defaultValue="@null" />

</fragment>
```

Generated Arguments

Create using constructor

```
// Create instance given a contactId : String
data class ContactDetailFragmentArgs(val contactId: String? = null) : NavArgs {

    // Output to a bundle
    fun toBundle(): Bundle { ... }

    companion object {

        // Create instance from a Bundle
        @JvmStatic
        fun fromBundle(bundle: Bundle): ContactDetailFragmentArgs { ... }
    }
}
```

Navigation Graph

```
<fragment android:id="@+id/contact_detail"
          android:name=".ContactDetailFragment">

    <argument android:name="contactId"
              app:argType="string"
              app:nullable="true"
              app:defaultValue="@null" />

</fragment>
```

Accessing Args in Destination

```
val args = ContactDetailFragmentArgs.fromBundle(requireArguments())

– or –

val args: ContactDetailFragmentArgs by navArgs()
```

Args

```
// Create instance given a contactId : String
data class ContactDetailFragmentArgs(val contactId: String? = null) : NavArgs {

    // Output to a bundle
    fun toBundle(): Bundle { ... }

    companion object {

        // Create instance from a Bundle
        @JvmStatic
        fun fromBundle(bundle: Bundle): ContactDetailFragmentArgs { ... }
    }
}
```

Navigation Graph

```
<fragment android:id="@+id/contact_detail"
          android:name=".ContactDetailFragment">

        <argument android:name="contactId"
                  app:argType="string"
                  app:nullable="true"
                  app:defaultValue="@null" />

</fragment>
```

Generated Arguments + Directions

```
<navigation
    android:id="@+id/contacts_graph"
    app:startDestination="@id/contacts_list">

    <fragment android:id="@+id/contacts_list" android:name=".ContactsListFragment">

        <action
            android:id="@+id/to_contact_detail"
            app:destination="@id/contact_detail" />

   </fragment>

    <fragment android:id=“@+id/contact_detail"
              android:name=".ContactDetailFragment">

        <argument android:name=“contactId”
                  app:argType=“string"
                  app:nullable="true"
                  app:defaultValue="@null" />

    </fragment>
</navigation>
```

Generated Arguments + Directions

```
<navigation
    android:id="@+id/contacts_graph"
    app:startDestination="@id/contacts_list">

    <fragment android:id="@+id/contacts_list" android:name=".ContactsListFragment">

        <action
            android:id="@+id/to_contact_detail"
            app:destination="@id/contact_detail" />

    </fragment>

    <fragment android:id="@+id/contact_detail"
              android:name=".ContactDetailFragment">

        <argument android:name="contactId"
                  app:argType="string"
                  app:nullable="true"
                  app:defaultValue="@null" />

    </fragment>
</navigation>
```

Generated Directions

```
class ContactsListFragmentDirections private constructor() {

    private data class ToContactDetail(val contactId: String? = null) : NavDirections {

        override fun getActionId(): Int = R.id.to_contact_detail

        override fun getArguments(): Bundle { ... }
    }

    companion object {
        fun toContactDetail(contactId: String? = null): NavDirections =
                ToContactDetail(contactId)
    }
}
```

Navigation Graph

```
<fragment android:id="@+id/contacts_list"
      android:name=".ContactsListFragment">

    <action
        android:id="@+id/to_contact_detail"
        app:destination="@id/contact_detail" />

</fragment>
```

Generated Directions

```
class ContactsListFragmentDirections private constructor() {

    private data class ToContactDetail(val contactId: String? = null) : NavDirections {

        override fun getActionId(): Int = R.id.to_contact_detail

        override fun getArguments(): Bundle { ... }
    }

    companion object {
        fun toContactDetail(contactId: String? = null): NavDirections =
                ToContactDetail(contactId)
    }
}
```

Only one way to construct an instance

Navigation Graph

```
<fragment android:id="@+id/contacts_list"
      android:name=".ContactsListFragment">

    <action
        android:id="@+id/to_contact_detail"
        app:destination="@id/contact_detail" />

</fragment>
```

Navigating with Directions

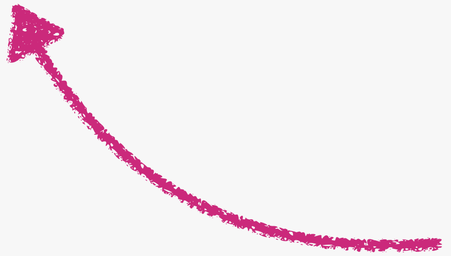
```
val directions =
        ContactsListFragmentDirections
            .toContactDetail(contact.id)

findNavController().navigate(directions)
```

Navigating with Directions


```
val directions =
        ContactsListFragmentDirections
            .toContactDetail(contact.id)

findNavController().navigate(directions)
```

Navigating with Directions

```
val directions =
        ContactsListFragmentDirections
            .toContactDetail(contact.id)

findNavController().navigate(
            toContactDetail(contact.id))
```



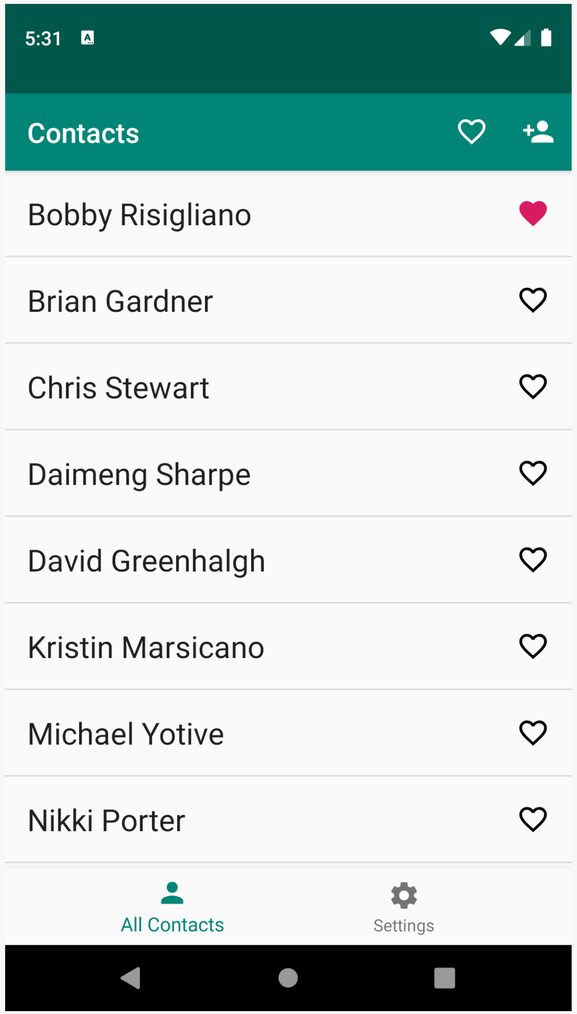
Declare Inline

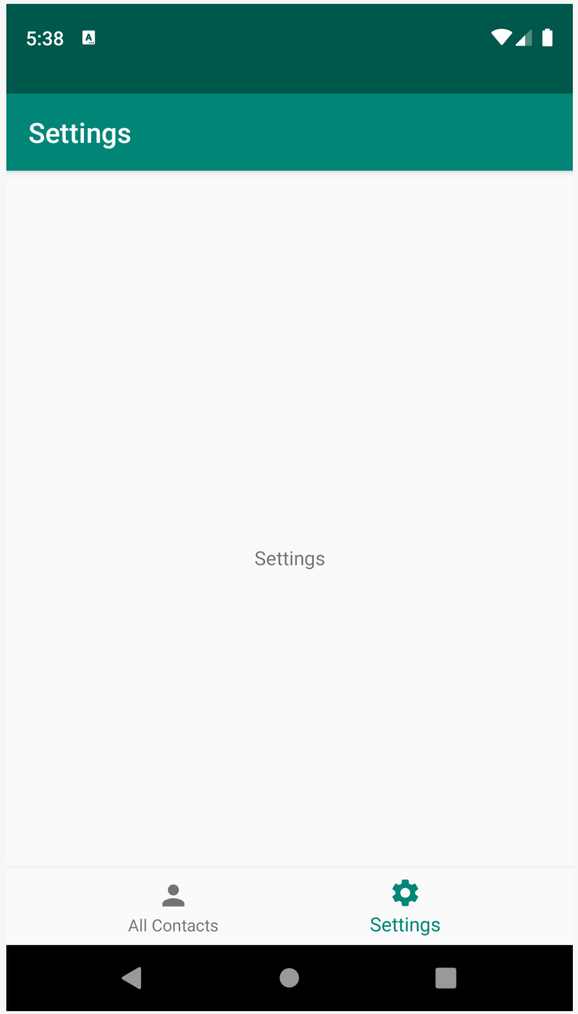
Lab 1: Getting Started

Lab 2: Top Level Navigation + Testing

Top-Level Navigation

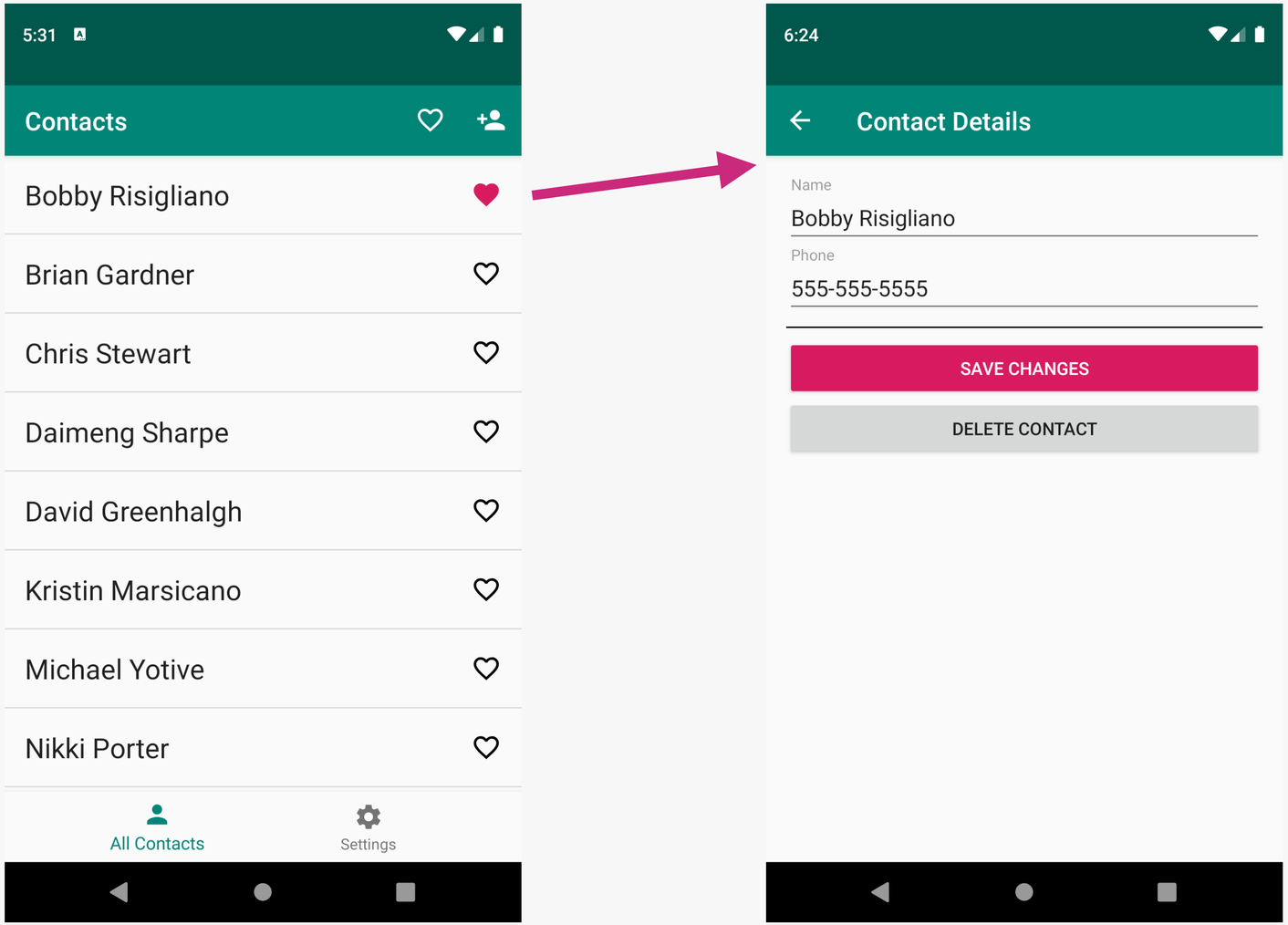
TOP LEVEL NAVIGATION





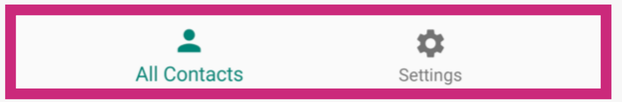
Bottom NavigationView

TOP LEVEL NAVIGATION



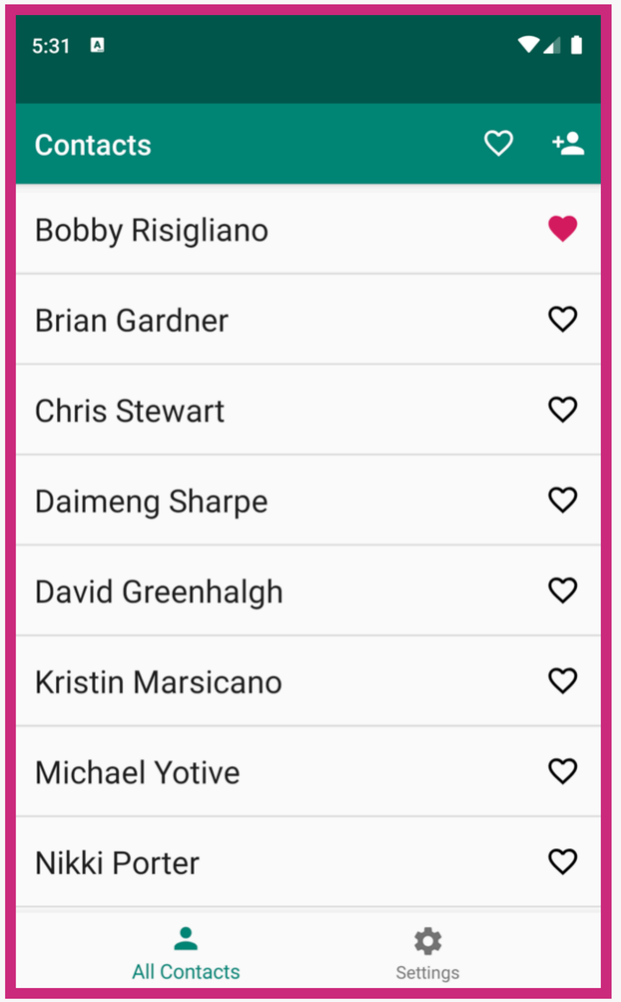
Add a new menu resource

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">

    <item
            android:id="@+id/contacts_list"
            android:icon="@drawable/ic_contact"
            android:title="@string/title_home"/>

    <item
            android:id="@+id/settings"
            android:icon="@drawable/ic_settings"
            android:title="@string/title_settings"/>

</menu>
```



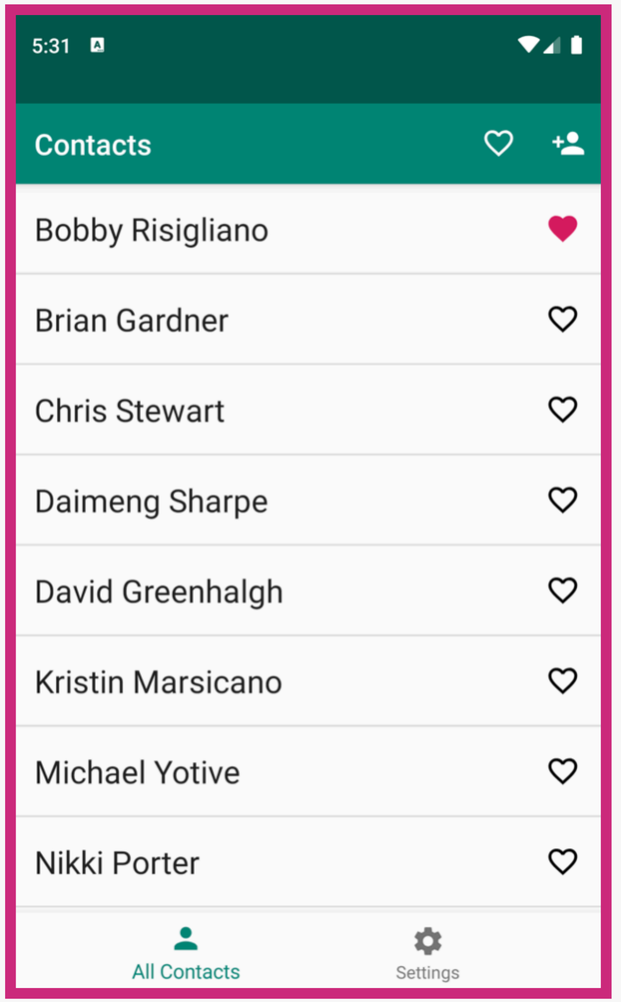
Add to Activity Layout

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout...>

    <fragment android:id="@+id/nav_host" .../>

    <com.google.android.material.bottomnavigation.BottomNavigationView
            android:id="@+id/bottom_nav_view"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginEnd="0dp"
            android:layout_marginStart="0dp"
            android:background="?android:attr/windowBackground"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintRight_toRightOf="parent"
            app:menu="@menu/bottom_nav_menu"/>

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



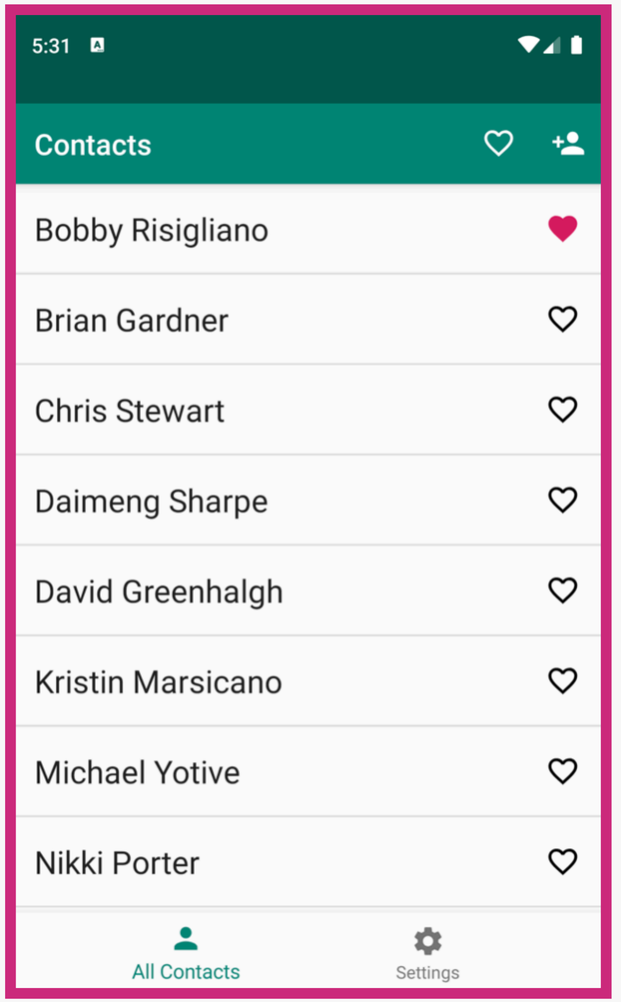
Navigation UI

```
class MainActivity : AppCompatActivity() {

    lateinit var navController: NavController
    lateinit var bottomNavigationView: BottomNavigationView

    override fun onCreate(savedInstanceState: Bundle?) {
        ...

        navController = findNavController(R.id.nav_host)
        bottomNavigationView = findViewById(R.id.bottom_nav_view)
        bottomNavigationView.setupWithNavController(navController)

    }

}
```



Navigation UI

```
class MainActivity : AppCompatActivity() {

    lateinit var navController: NavController
    lateinit var bottomNavigationView: BottomNavigationView

    override fun onCreate(savedInstanceState: Bundle?) {
        ...

        navController = findNavController(R.id.nav_host)
        bottomNavigationView = findViewById(R.id.bottom_nav_view)
        bottomNavigationView.setupWithNavController(navController)

    }

}
```




```
"androidx.navigation:navigation-ui-ktx:$version"
```

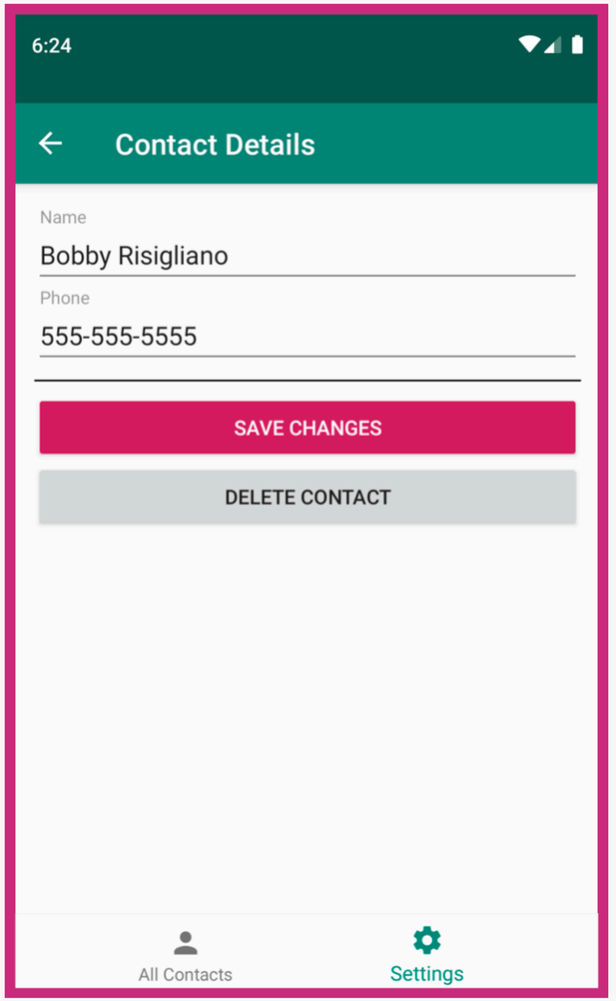
Add a new menu resource

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu>
   <item android:id="@+id/contacts_list"/>

   <item android:id="@+id/settings"/>

</menu>
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<navigation android:id="@+id/contacts_graph"
            app:startDestination="@id/contacts_list">

   <fragment android:id="@+id/contacts_list".../>

   <fragment android:id="@+id/contact_detail" .../>

   <fragment android:id="@+id/settings".../>

</navigation>
```

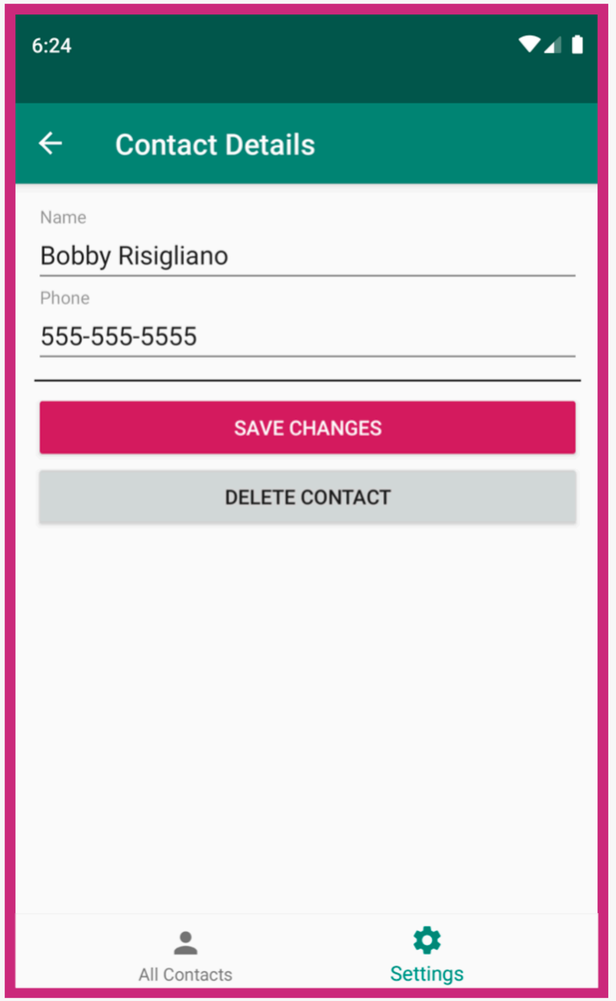
Add Up Button

```
class MainActivity : AppCompatActivity() {

    lateinit var navController: NavController
    lateinit var bottomNavigationView: BottomNavigationView

    override fun onCreate(savedInstanceState: Bundle?) {
        ...

        navController = findNavController(R.id.nav_host)
        bottomNavigationView = findViewById(R.id.bottom_nav_view)
        bottomNavigationView.setupWithNavController(navController)

    }

}
```



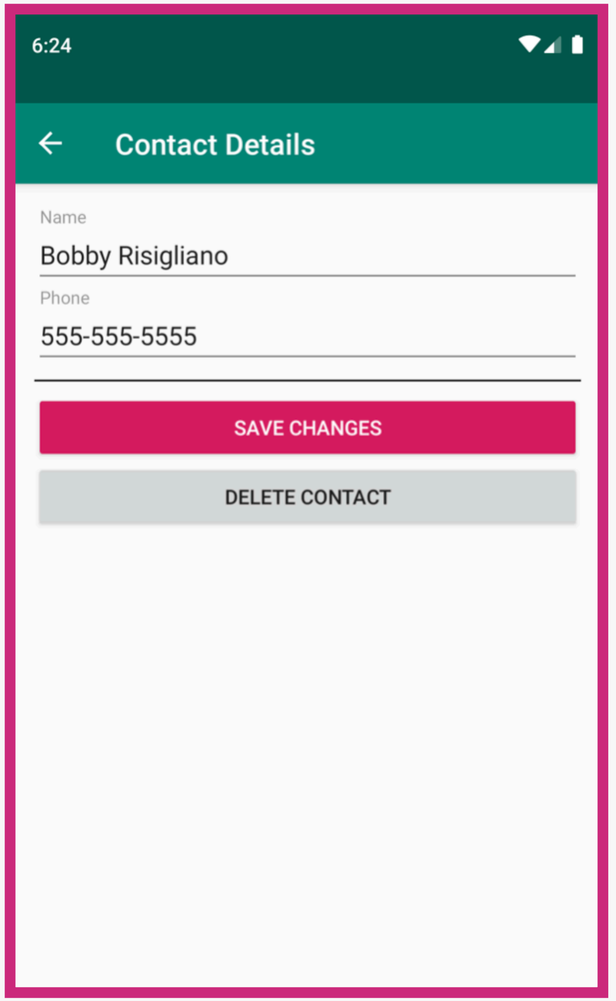
Add Up Button

```
class MainActivity : AppCompatActivity() {

    lateinit var navController: NavController
    lateinit var bottomNavigationView: BottomNavigationView

    override fun onCreate(savedInstanceState: Bundle?) {
        ...

        navController = findNavController(R.id.nav_host)
        bottomNavigationView = findViewById(R.id.bottom_nav_view)
        bottomNavigationView.setupWithNavController(navController)
        setupActionBarWithNavController(navController)

    }

}

override fun onSupportNavigateUp(): Boolean {
    return navController.navigateUp() || super.onSupportNavigateUp()}
```



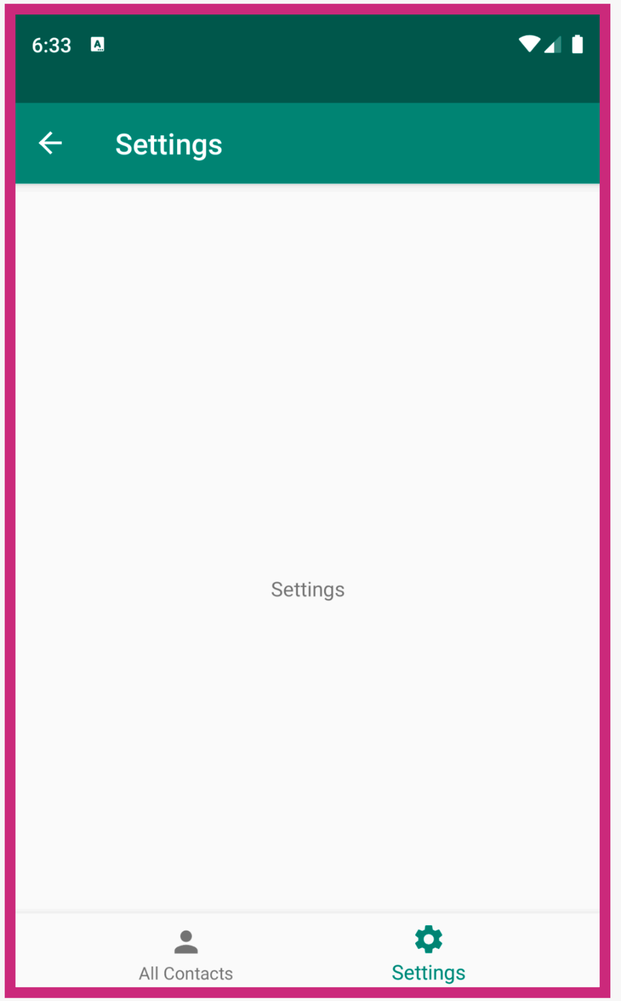
Hide Bottom Nav on Details

```
class MainActivity : AppCompatActivity() {

    lateinit var navController: NavController
    lateinit var bottomNavigationView: BottomNavigationView

    override fun onCreate(savedInstanceState: Bundle?) {
        ...

        navController.addOnDestinationChangedListener {
          navController, newDestination, arguments ->
            when (newDestination.id) {
               R.id.contact_detail ->
                   bottomNavigationView.visibility = View.GONE
               else ->
                   bottomNavigationView.visibility = View.VISIBLE
            }
        }

    }
}
...
```



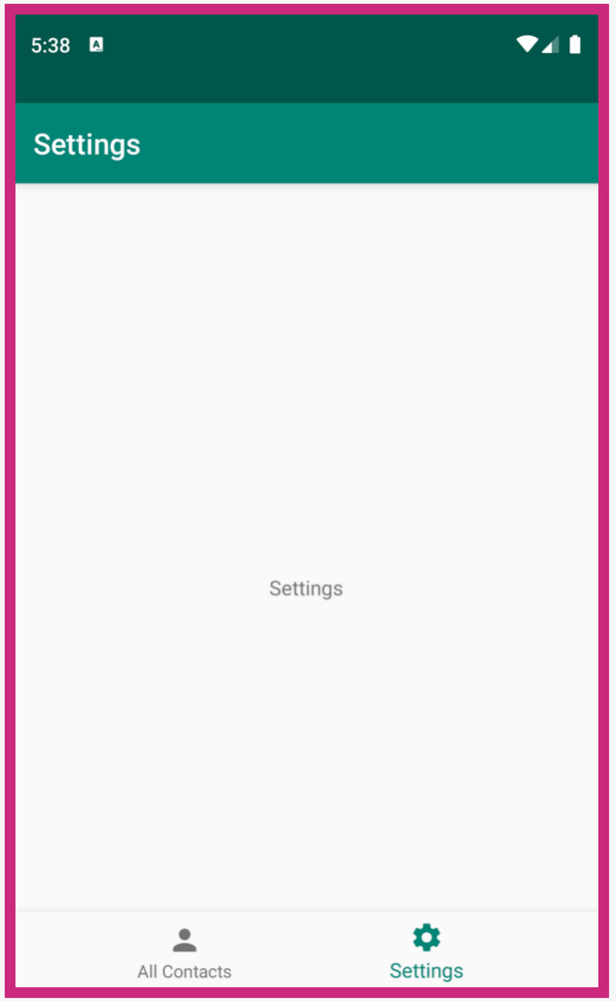
Add Addit'l TLDs

```
class MainActivity : AppCompatActivity() {

    lateinit var navController: NavController
    lateinit var bottomNavigationView: BottomNavigationView

    override fun onCreate(savedInstanceState: Bundle?) {
        ...

        navController = findNavController(R.id.nav_host)
        bottomNavigationView = findViewById(R.id.bottom_nav_view)
        bottomNavigationView.setupWithNavController(navController)

        setupActionBarWithNavController(navController)
    }

}
```



Add Addit'l TLDs

```
class MainActivity : AppCompatActivity() {

    lateinit var navController: NavController
    lateinit var appBarConfiguration: AppBarConfiguration
    lateinit var bottomNavigationView: BottomNavigationView

    override fun onCreate(savedInstanceState: Bundle?) {
        ...

        navController = findNavController(R.id.nav_host)
        bottomNavigationView = findViewById(R.id.bottom_nav_view)
        bottomNavigationView.setupWithNavController(navController)

        appBarConfiguration = AppBarConfiguration(
                                    setOf(R.id.contacts_list, R.id.settings))
        setupActionBarWithNavController(navController, appBarConfiguration)

    }

}
```



Testing Navigation

Lab Testing Scenarios

Navigation to Contact Details

- ***When*** a contact is clicked on the contact_list
- ***Then*** NavController.navigate(toContactDetail(...)) is invoked with the corresponding contactId.

Lab Testing Scenarios

Navigation to Contact Details

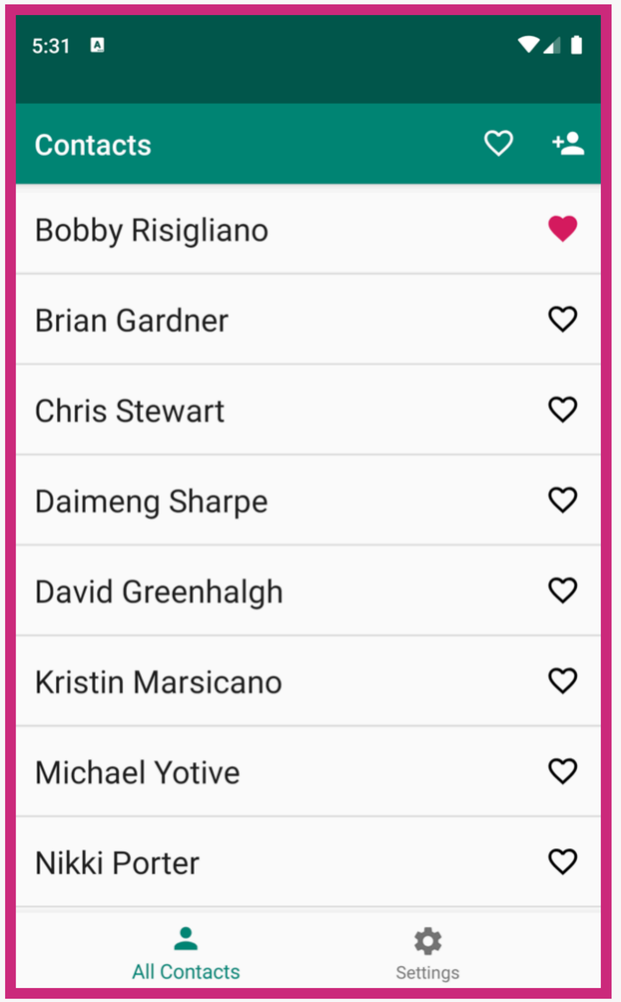
- ***When*** a contact is clicked on the contact_list
- ***Then*** NavController.navigate(toContactDetail(...)) is invoked with the corresponding contactId.

Contact Details Launches with Args + Pops Back on Save

- ***When*** the contact_detail is launched with a valid contactId.
- ***Then*** the contact details form populates with the expected information
  and NavController.popBactStack() is invoked after the user clicks the save.

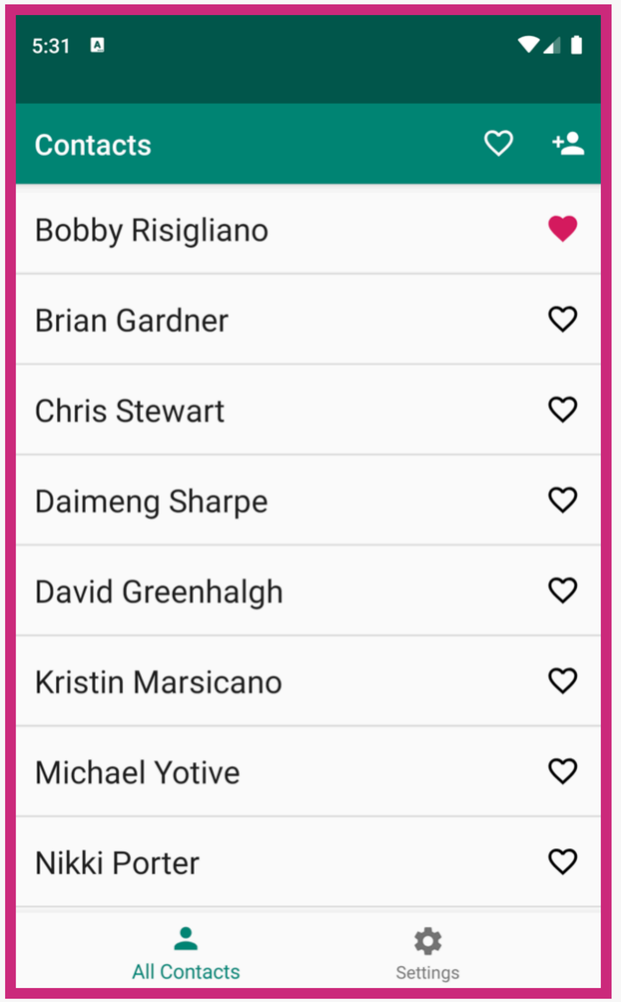
Mock NavController

```
@Test
fun testNavigateToContactDetails() {

    // 1. Create a mock instance of NavController
    val mockNavController = mock(NavController::class.java)









}
```



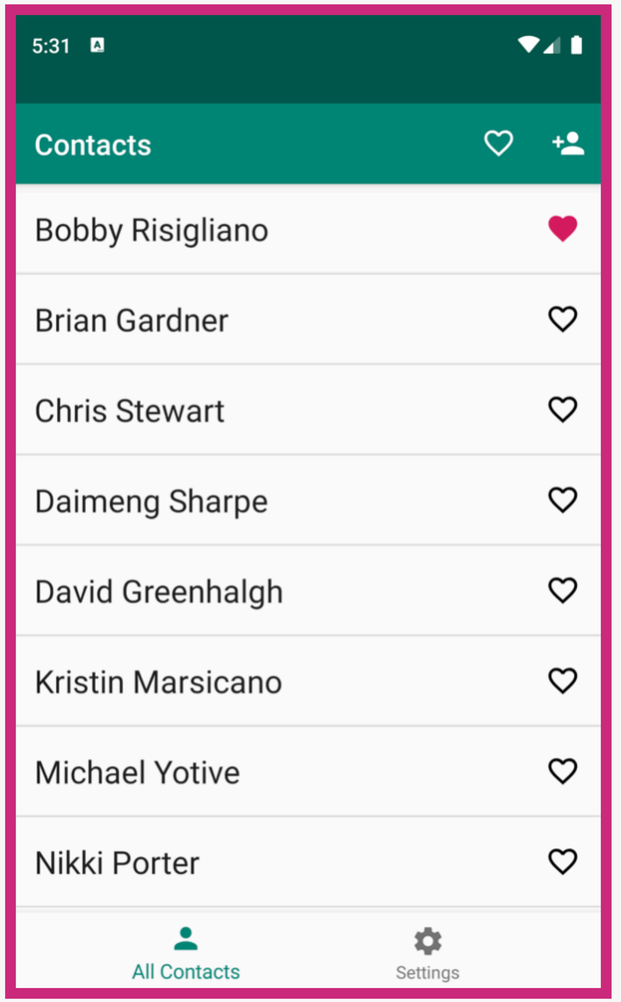
Mock NavController

```
@Test
fun testNavigateToContactDetails() {

    // 1. Create a mock instance of NavController
    val mockNavController = mock(NavController::class.java)

    // 2. Create FragmentScenario instance
    val fragmentScenario = launchFragmentInContainer<ContactsListFragment>()









}
```



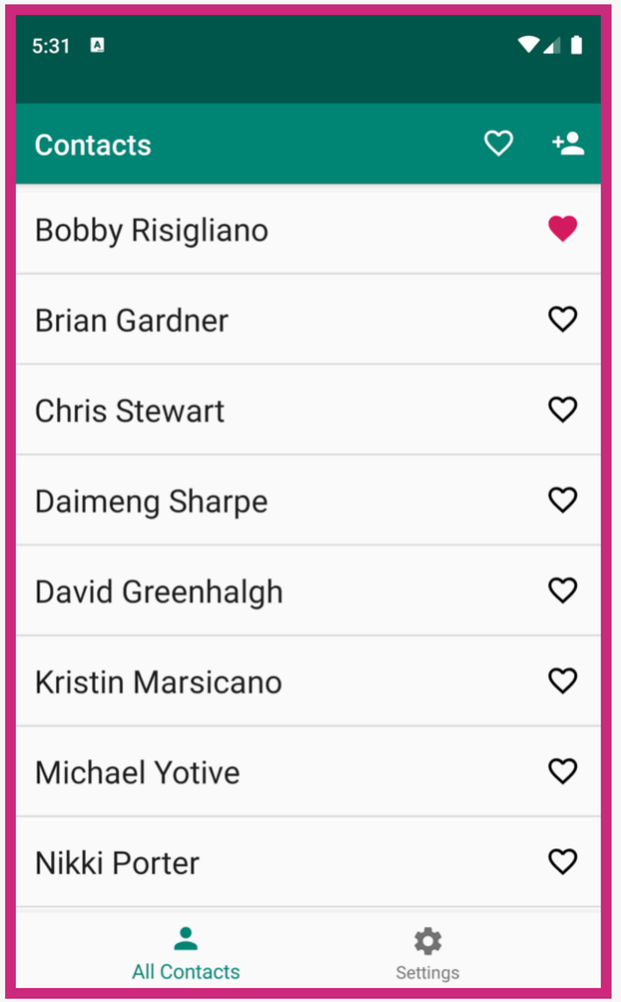
Mock NavController

```
@Test
fun testNavigateToContactDetails() {

    // 1. Create a mock instance of NavController
    val mockNavController = mock(NavController::class.java)

    // 2. Create FragmentScenario instance
    val fragmentScenario = launchFragmentInContainer<ContactsListFragment>()

    // 3. Wait for the fragment to become RESUMED.
    fragmentScenario.onFragment { fragment ->




    }




}
```



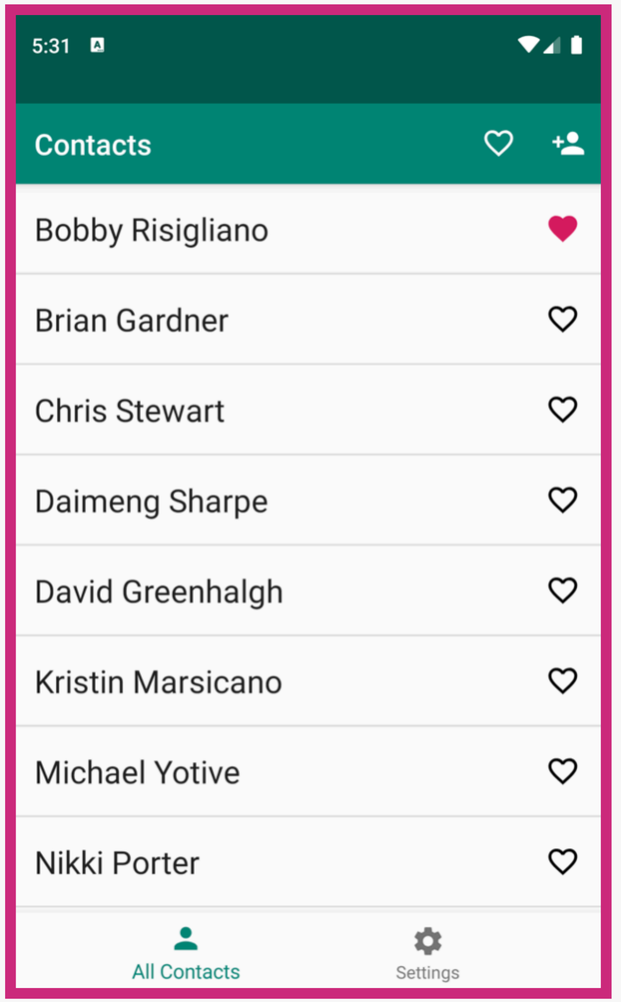
Mock NavController

```
@Test
fun testNavigateToContactDetails() {

    // 1. Create a mock instance of NavController
    val mockNavController = mock(NavController::class.java)

    // 2. Create FragmentScenario instance
    val fragmentScenario = launchFragmentInContainer<ContactsListFragment>()

    // 3. Wait for the fragment to become RESUMED.
    fragmentScenario.onFragment { fragment ->

        // 4. Replace the NavController with a mock instance.
        Navigation.setViewNavController(
            fragment.requireView(), mockNavController
        )
    }




}
```



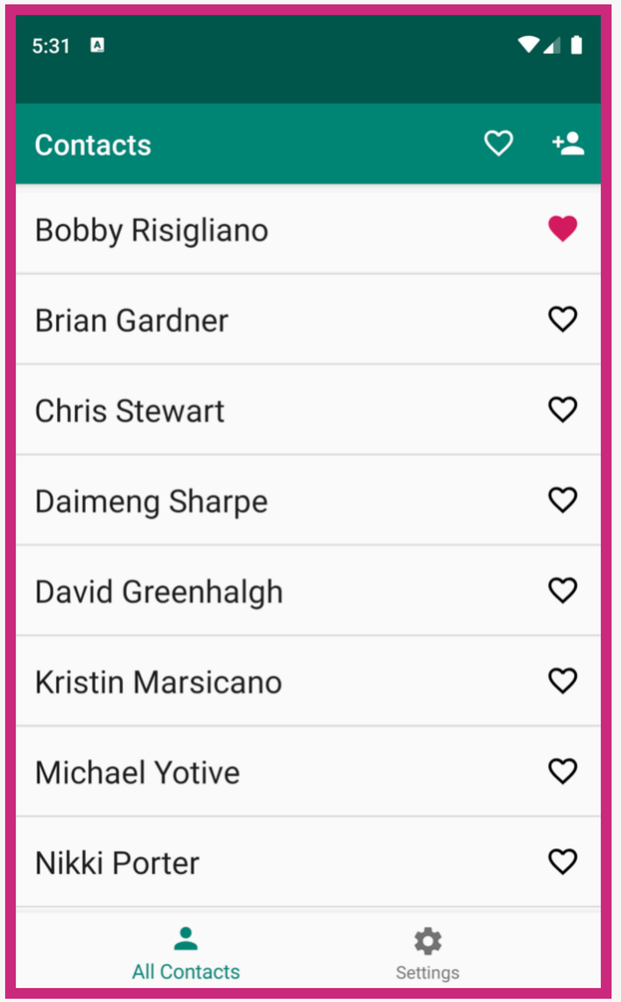
Mock NavController

```
@Test
fun testNavigateToContactDetails() {

    // 1. Create a mock instance of NavController
    val mockNavController = mock(NavController::class.java)

    // 2. Create FragmentScenario instance
    val fragmentScenario = launchFragmentInContainer<ContactsListFragment>()

    // 3. Wait for the fragment to become RESUMED.
    fragmentScenario.onFragment { fragment ->

        // 4. Replace the NavController with a mock instance.
        Navigation.setViewNavController(
            fragment.requireView(), mockNavController
        )
    }

    // 5. Perform an Action with Espresso
   onView(withId(R.id.rv_contacts_list))
        .perform(actionOnItemAtPosition<ContactViewHolder>(0, click()));


}
```



Mock NavController

```
@Test
fun testNavigateToContactDetails() {

    // 1. Create a mock instance of NavController
    val mockNavController = mock(NavController::class.java)

    // 2. Create FragmentScenario instance
    val fragmentScenario = launchFragmentInContainer<ContactsListFragment>()

    // 3. Wait for the fragment to become RESUMED.
    fragmentScenario.onFragment { fragment ->

        // 4. Replace the NavController with a mock instance.
        Navigation.setViewNavController(
            fragment.requireView(), mockNavController
        )
    }

    // 5. Perform an Action with Espresso
   onView(withId(R.id.rv_contacts_list))
        .perform(actionOnItemAtPosition<ContactViewHolder>(0, click()));

    // 6. Verify interactions with the NavController
    verify(mockNavController).navigate(toContactDetail(expectedContact.id))
}
```



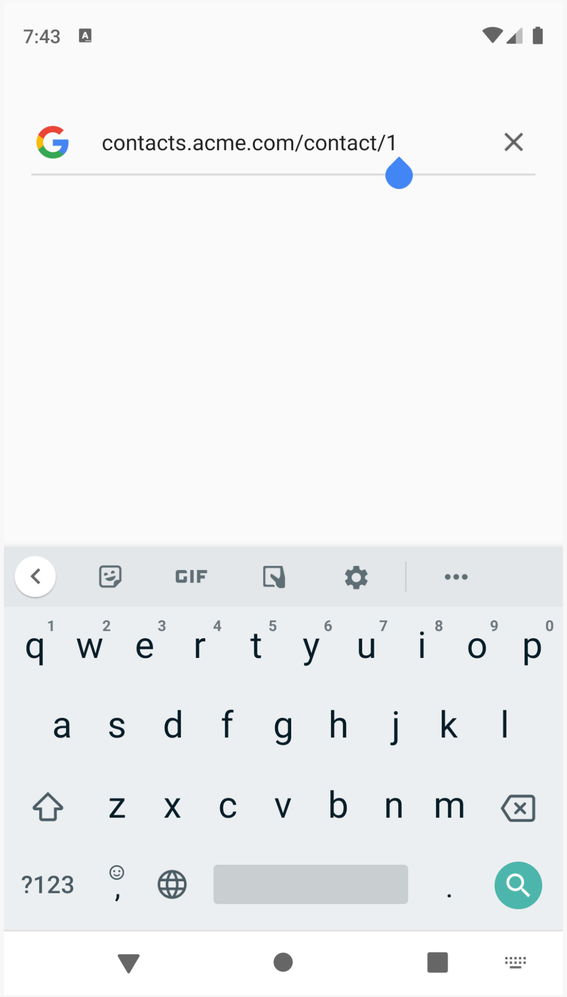
Lab 2: Top Level Navigation + Testing

Lab 3: Deep Linking + Conditional Navigation

Deep Linking

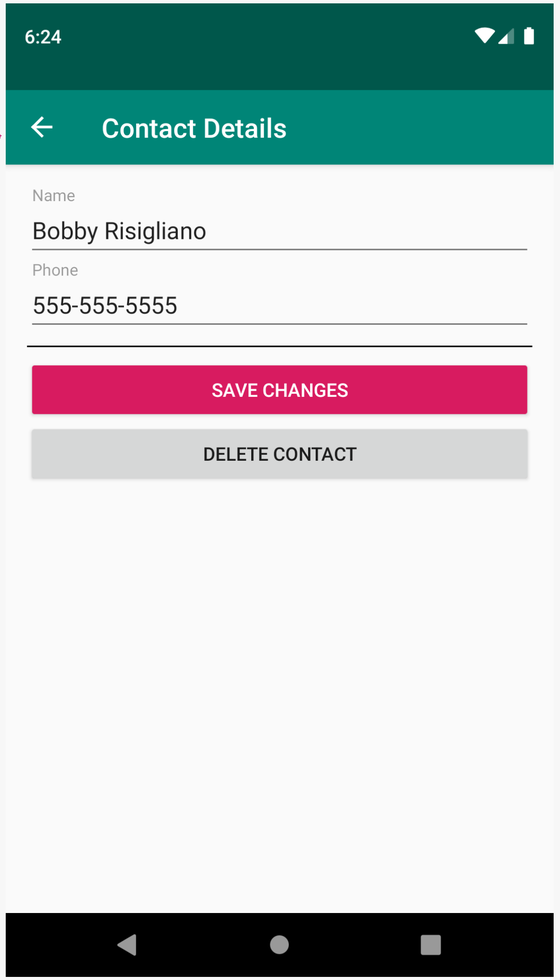
DEEP LINK

Implicit



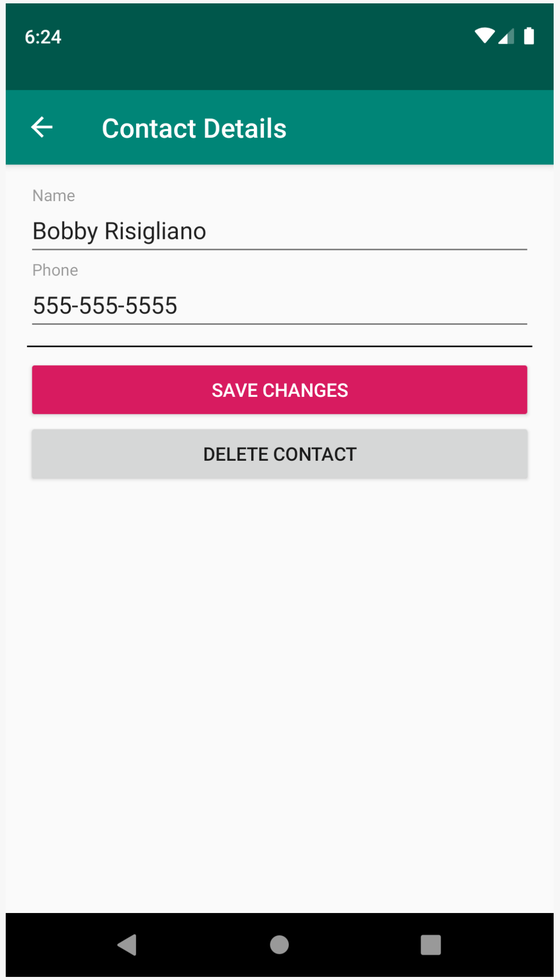
URI

contacts.acme.com/contact/{id}



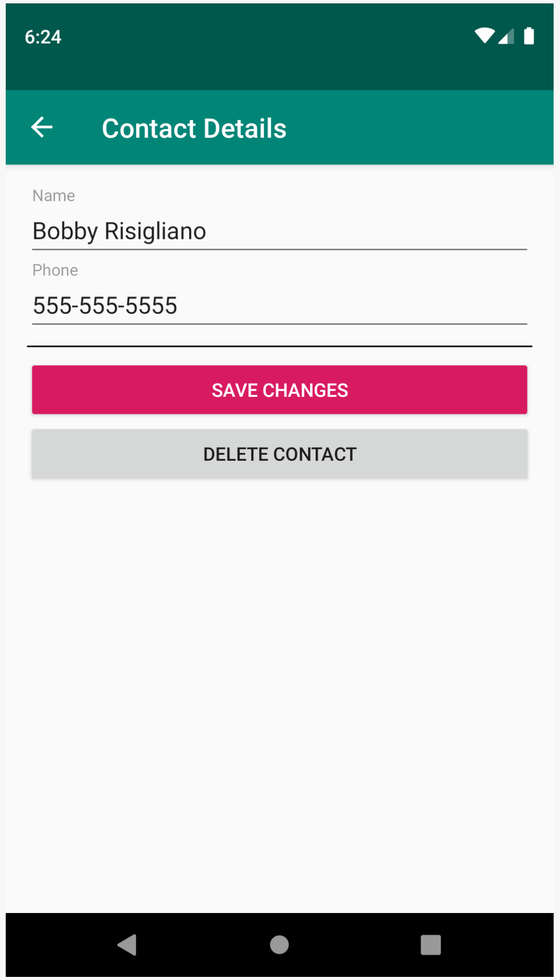
Implicit

```
<fragment android:id="@+id/contact_detail"
          android:name="com.acme.contacts.detail.ContactDetailFragment"
          tools:layout="@layout/fragment_contact_detail"
          android:label="Contact Details">

    <argument android:name="contactId"
              app:argType="string"
              app:nullable="true"
              android:defaultValue="@null"/>

    <deepLink
            android:id="@+id/deepLink"
            app:uri="contacts.acme.com/contact/{contactId}" />

</fragment>
```



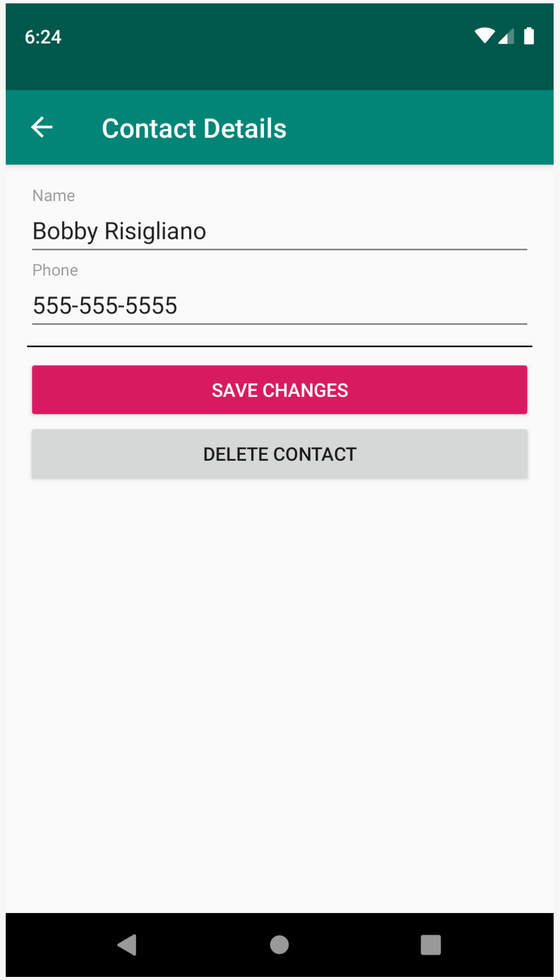
Implicit

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.acme.contacts">

   <application
        ...

        <activity android:name=".MainActivity">

            <nav-graph android:value="@navigation/contacts_graph" />

            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>

        </activity>

    </application>

</manifest>
```



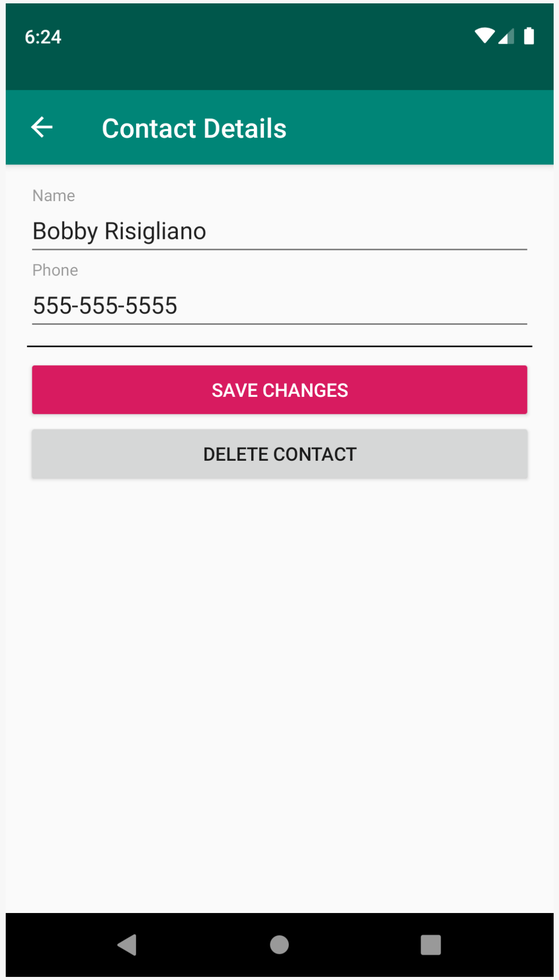
Implicit

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.acme.contacts">

   <application
        ...

        <activity android:name=".MainActivity">

            <intent-filter>
                <action android:name="android.intent.action.VIEW" />

                <category android:name="android.intent.category.DEFAULT" />
                <category android:name="android.intent.category.BROWSABLE" />

                <data android:scheme="http" />
                <data android:scheme="https" />
                <data android:host="contacts.acme.com" />
                <data android:pathPrefix="/contact/" />
            </intent-filter>
        ...
        </activity>

    </application>

</manifest>
```



On SetGraph

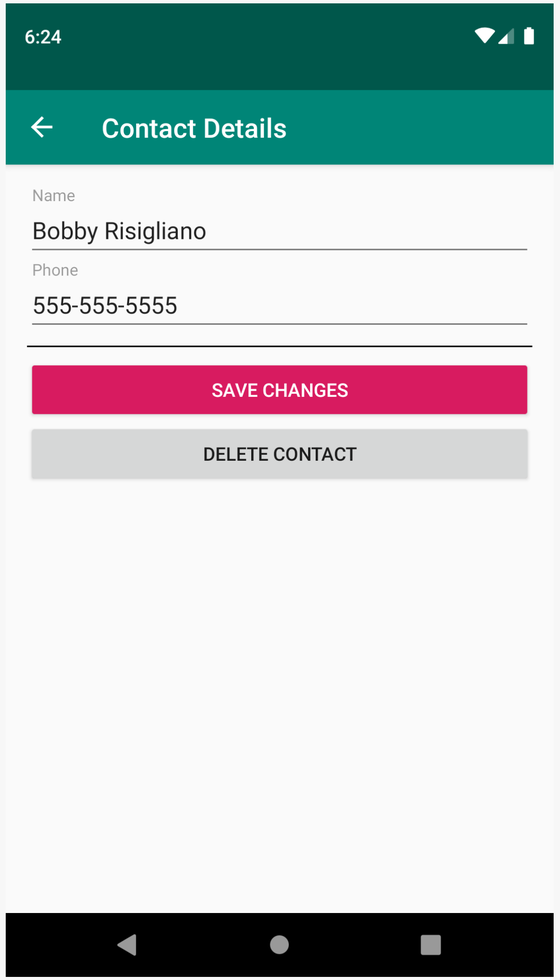
1. **Search Graph for best matching deep link**
   - URI exact matches preferred
   - By matching arguments



On SetGraph

1. **Search Graph for best matching deep link**
   - URI exact matches preferred
   - By matching arguments

2. **Build array of destination IDs to push on navigation stack**
   - R.id.contacts_list
   - R.id.contact_detail (includes self)

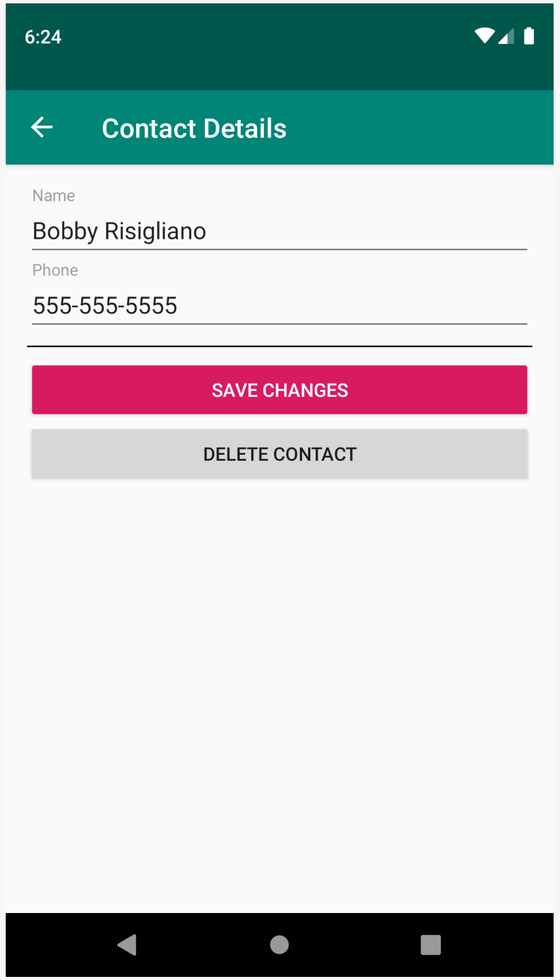


On SetGraph

1. **Search Graph for best matching deep link**
   - URI exact matches preferred
   - By matching arguments

2. **Build array of destination IDs to push on navigation stack**
   - R.id.contacts_list
   - R.id.contact_detail (includes self)

3. **Push the start and arrival destinations on the stack**
   - (If new task)

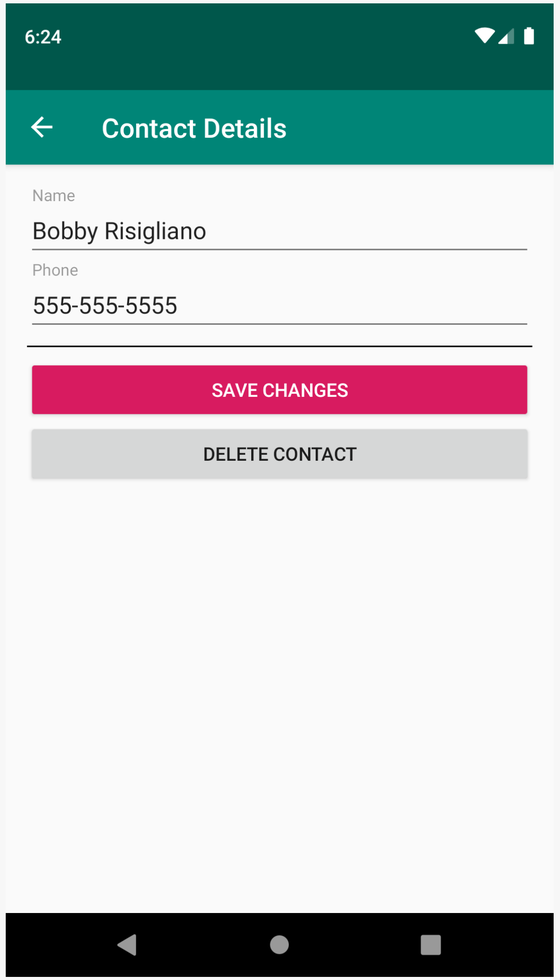


On SetGraph

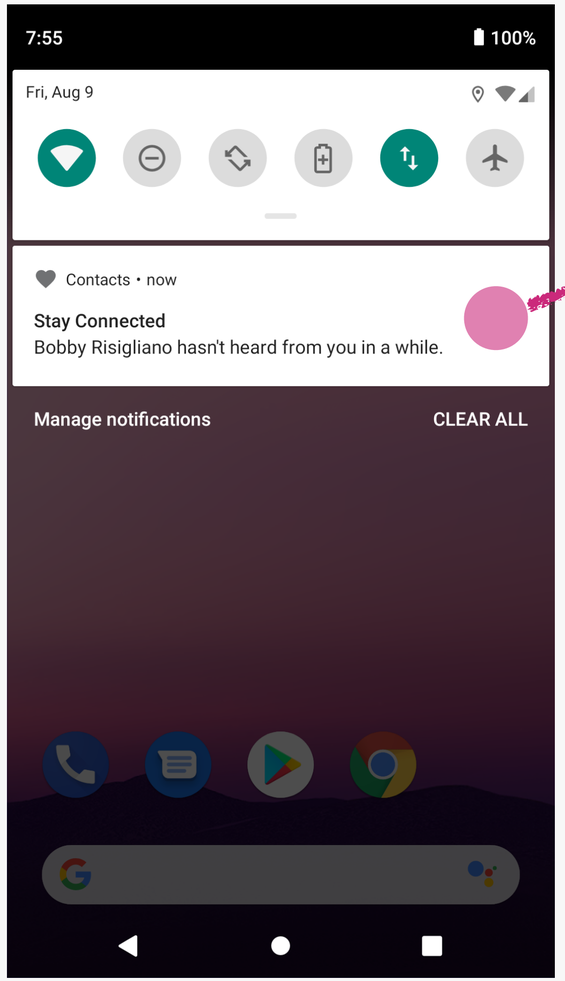
1. **Search Graph for best matching deep link**
   - URI exact matches preferred
   - By matching arguments

2. **Build array of destination IDs to push on navigation stack**
   - R.id.contacts_list
   - R.id.contact_detail (includes self)

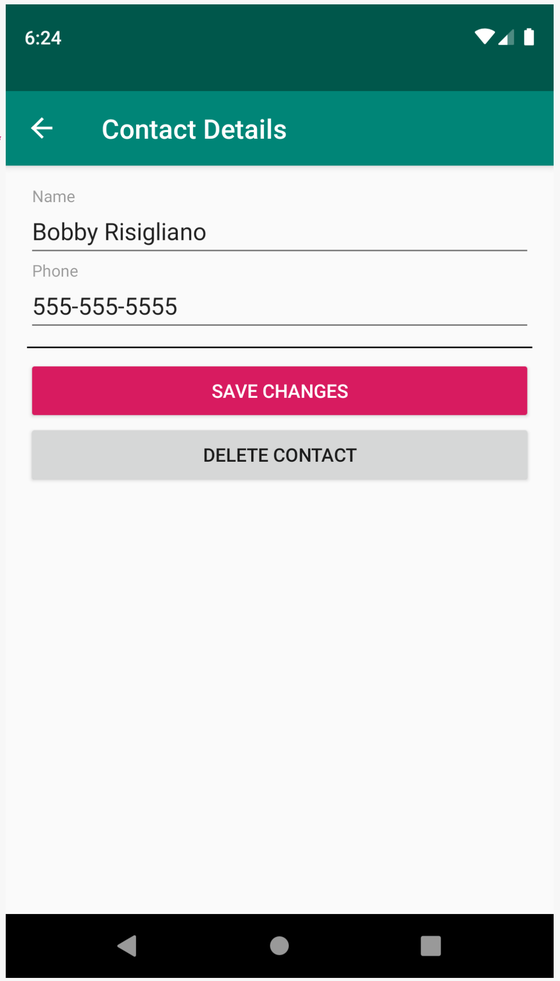
3. **Push last destination on the stack**
   - (If part of an existing task)



Explicit

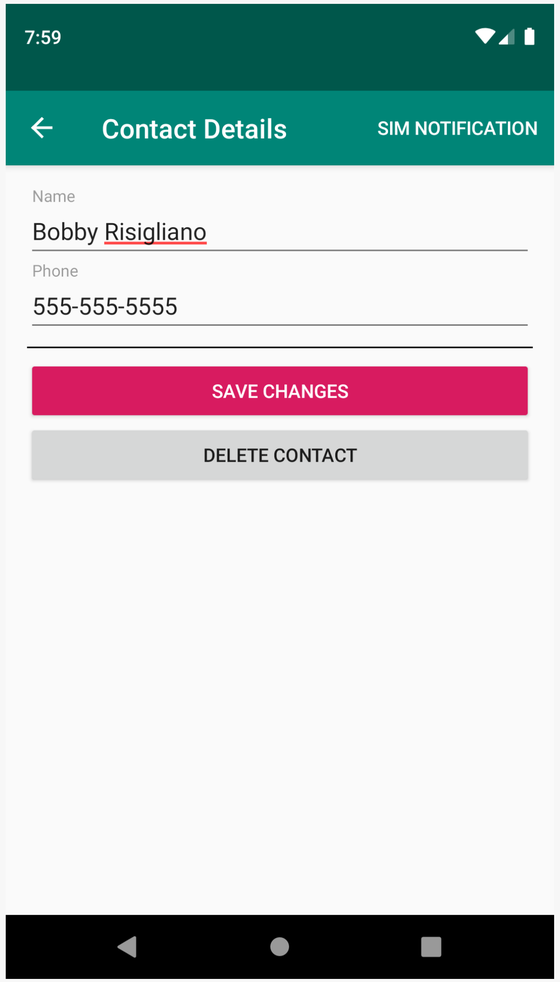


Launch via `PendingIntent`



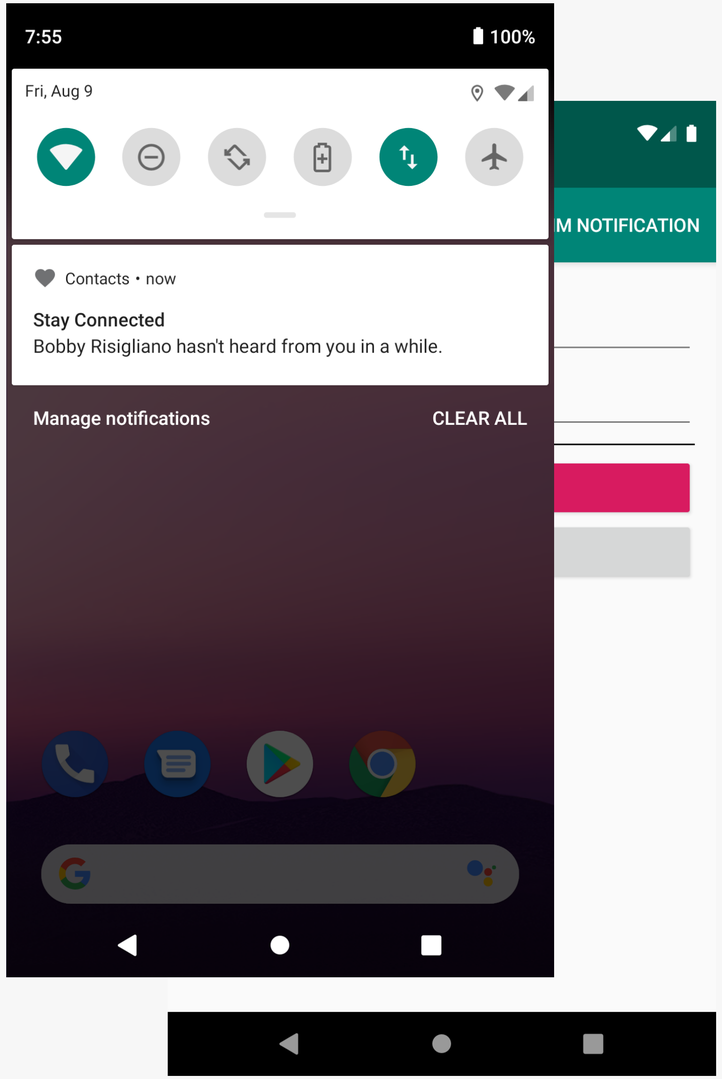
Create Notification

```
val pendingIntent = findNavController().createDeepLink()
    .setDestination(R.id.contact_detail)
    .setArguments(arguments)
    .createPendingIntent()

val notification = NotificationCompat.Builder(...)
            .setContentTitle(...)
            .setSmallIcon(...)
            .setContentText(...)
            .setContentIntent(pendingIntent)
            .setAutoCancel(true)
            .build()

val notificationManager = NotificationManagerCompat.from(...)
notificationManager.notify(0, notification)
```



Create a Pending Intent

```
val pendingIntent = findNavController().createDeepLink()
    .setDestination(R.id.contact_detail)
    .setArguments(arguments)
    .createPendingIntent()

val notification = NotificationCompat.Builder(...)
            .setContentTitle(...)
            .setSmallIcon(...)
            .setContentText(...)
            .setContentIntent(pendingIntent)
            .setAutoCancel(true)
            .build()

val notificationManager = NotificationManagerCompat.from(...)
notificationManager.notify(0, notification)
```

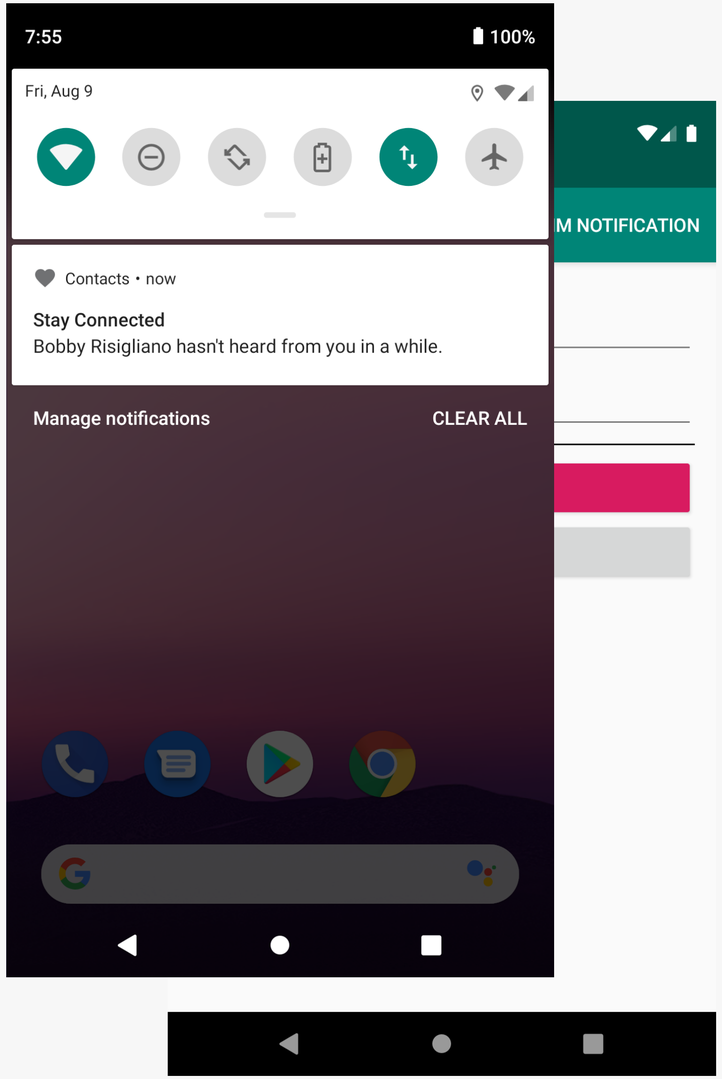


Explicit Deep Link Intent

1. **android-support-nav:controller:deepLinkIds**
   - R.id.contacts_list
   - R.id.contact_detail

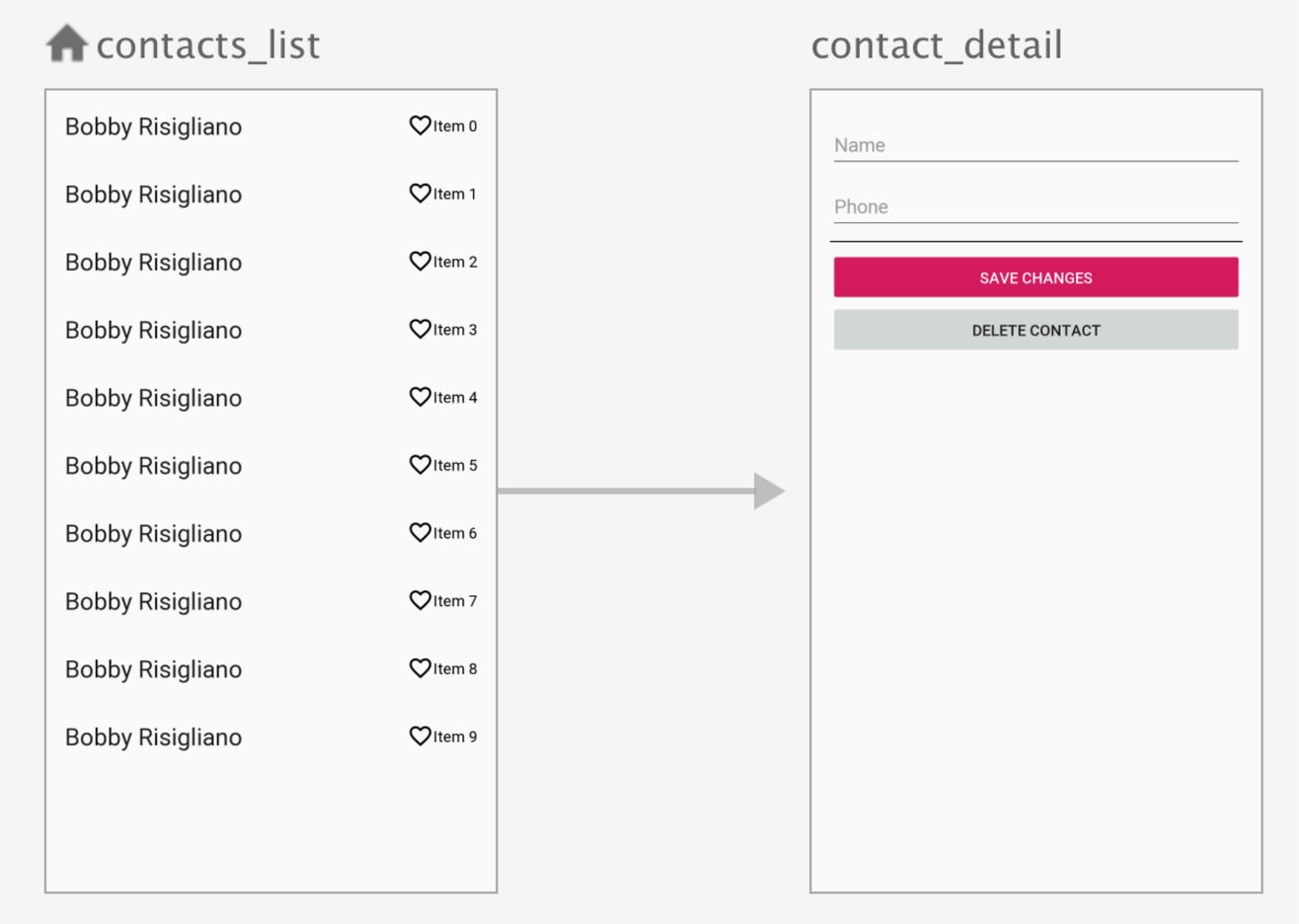
2. **android-support-nav:controller:deepLinkExtras**
   - contactId

3. **android-support-nav:controller:deepLinkIntent**
   - Intent that triggered the deep link



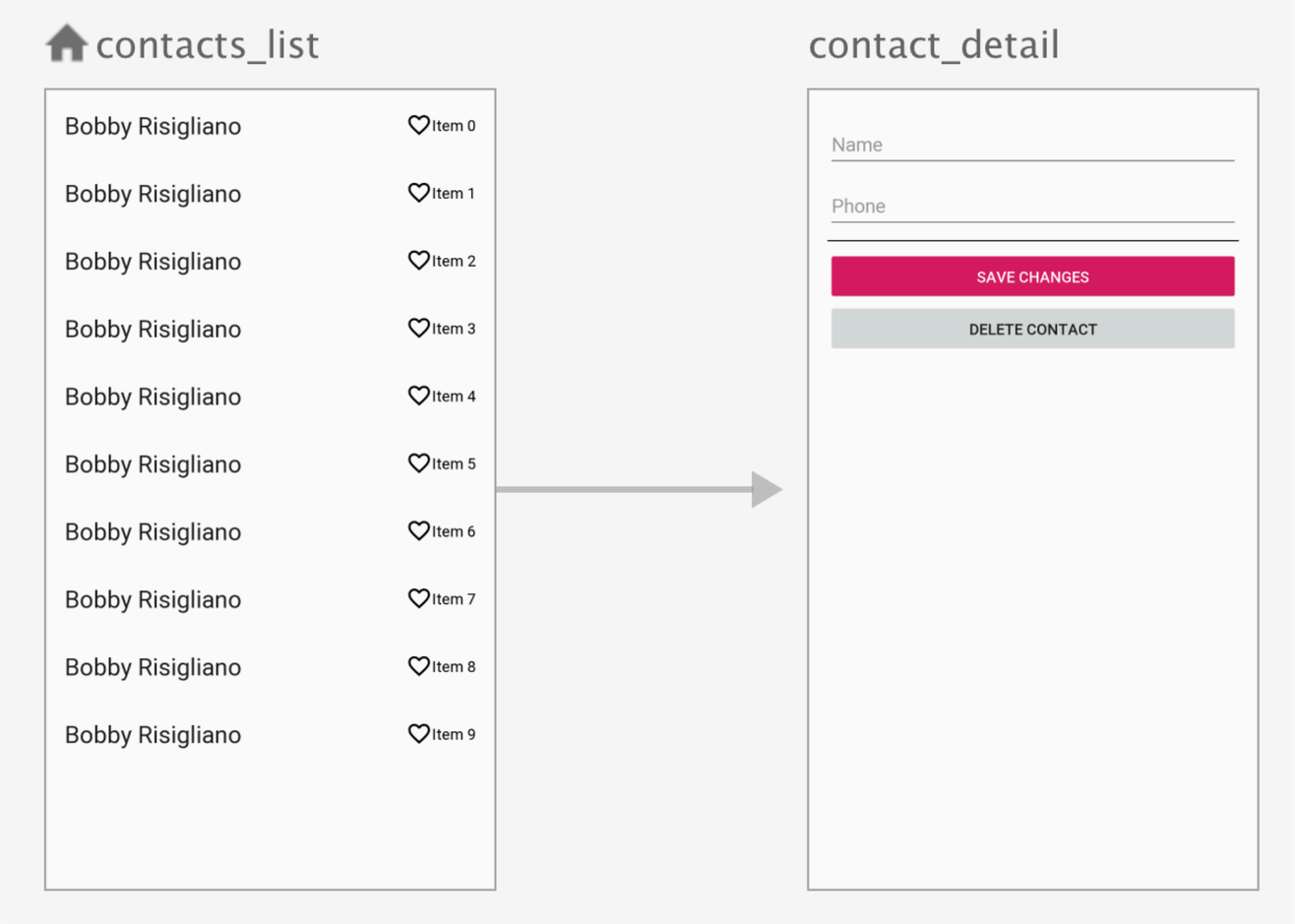
Conditional Navigation

CONDITIONAL NAVIGATION

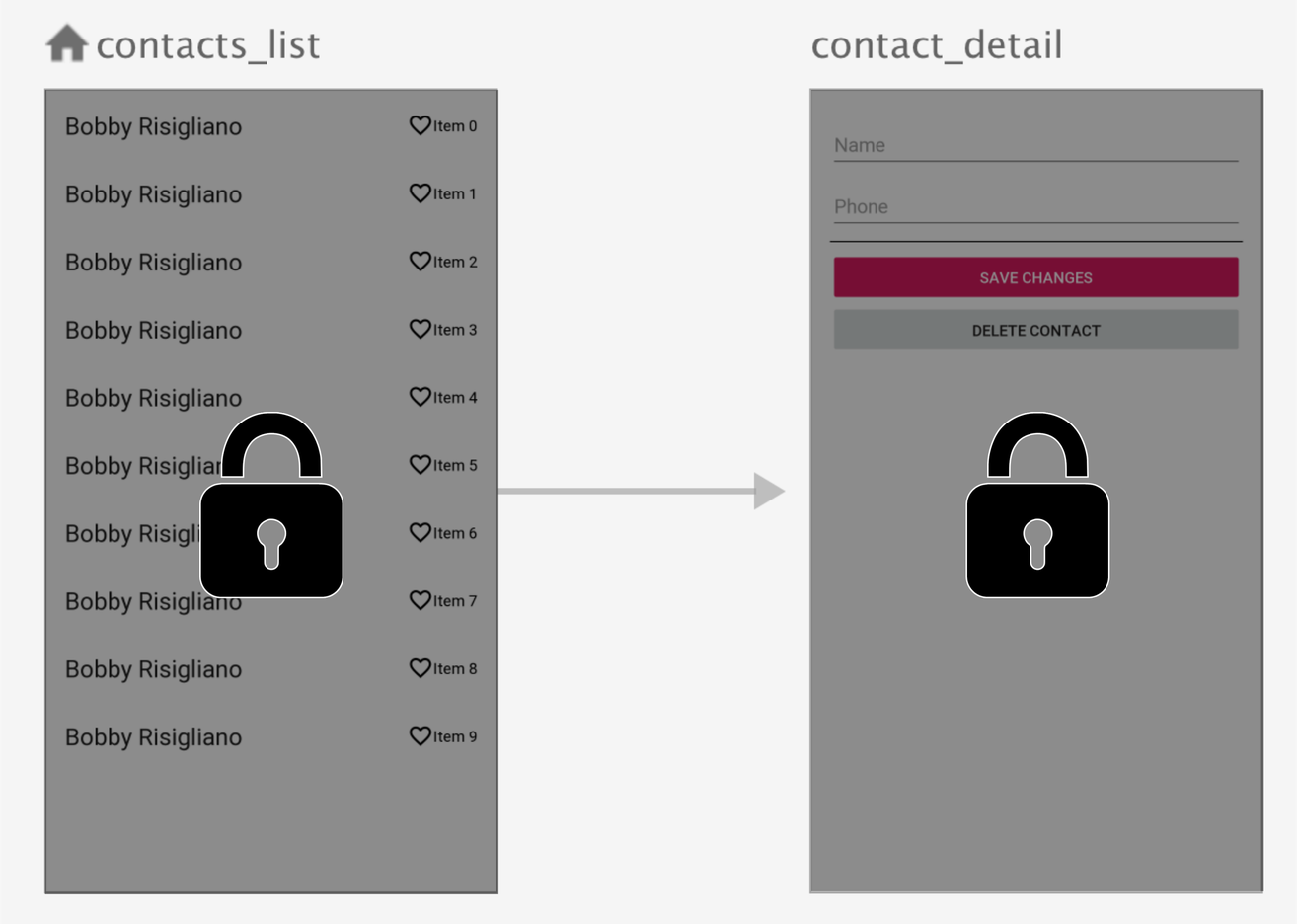


CONDITIONAL NAVIGATION

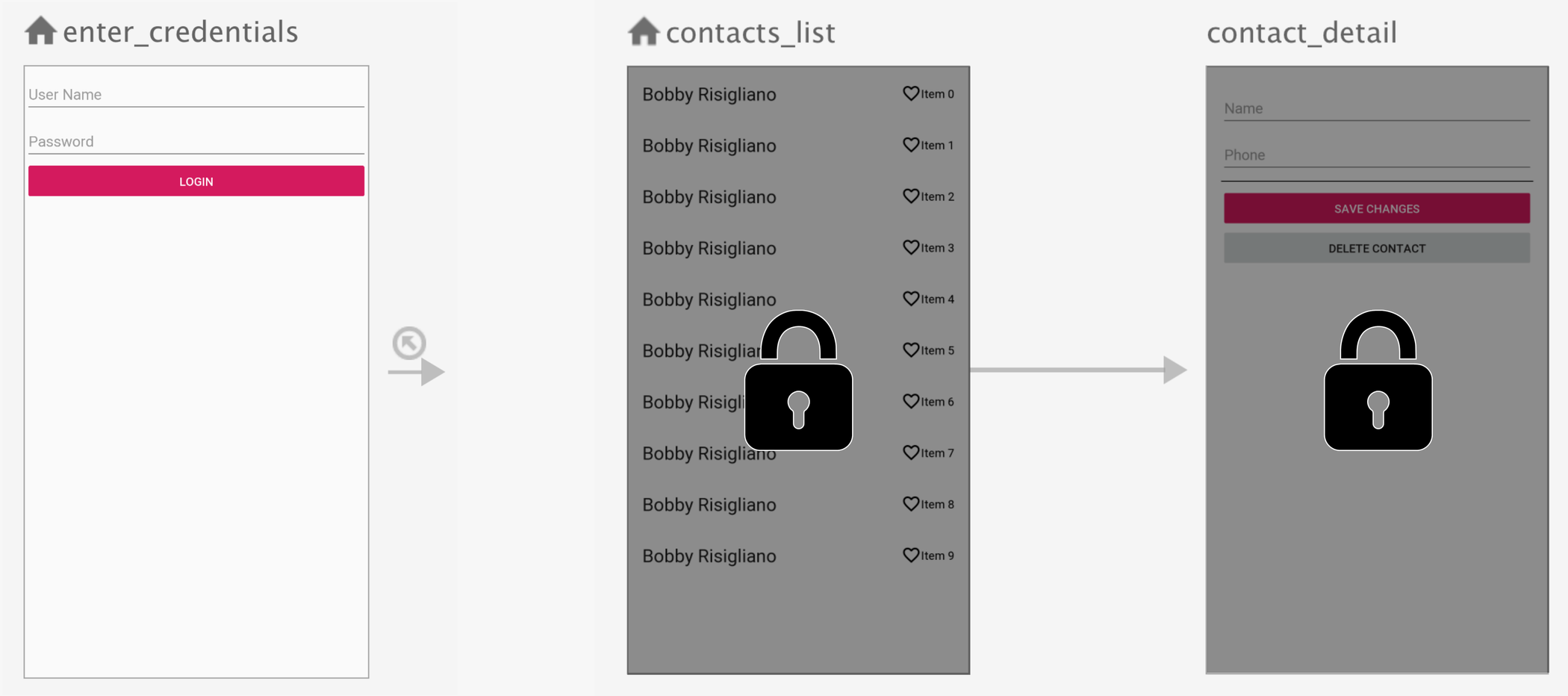
Fixed Starting Destination



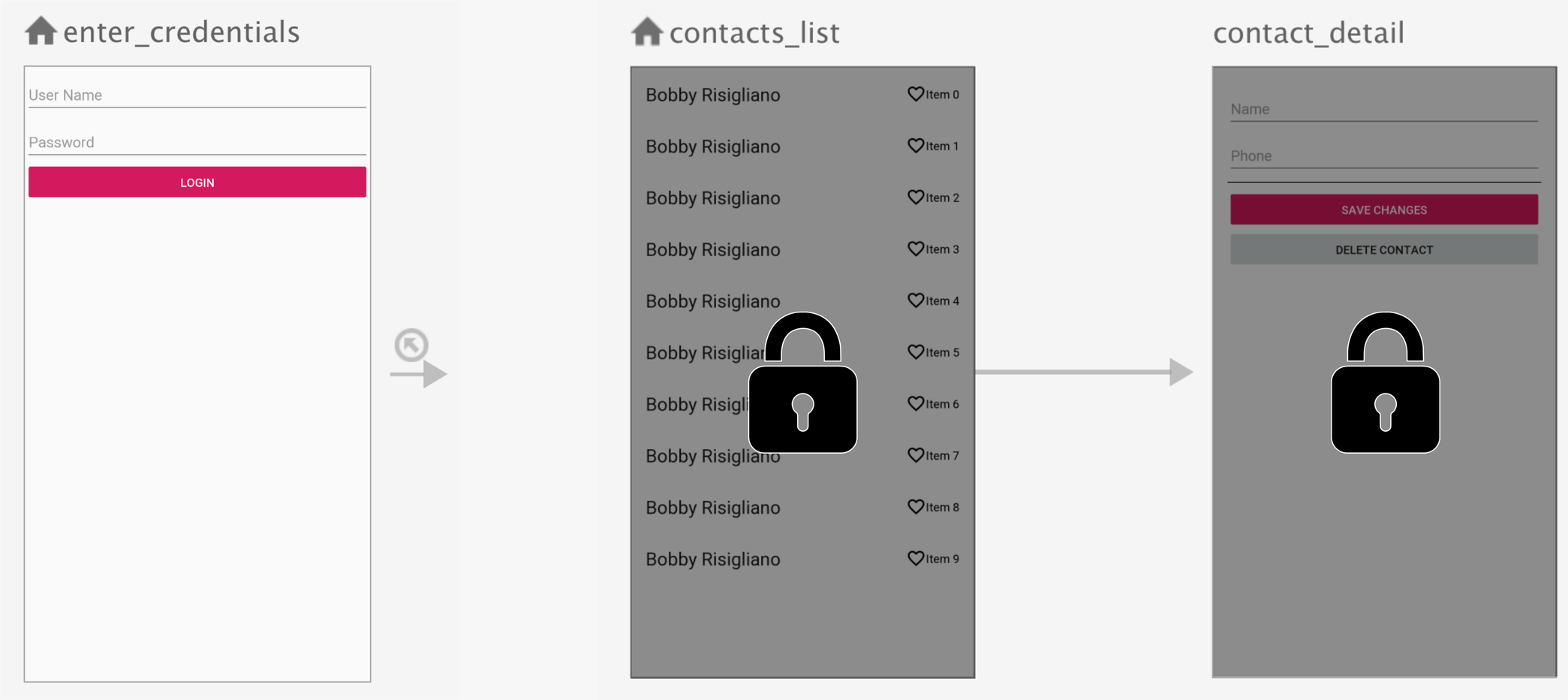
CONDITIONAL NAVIGATION



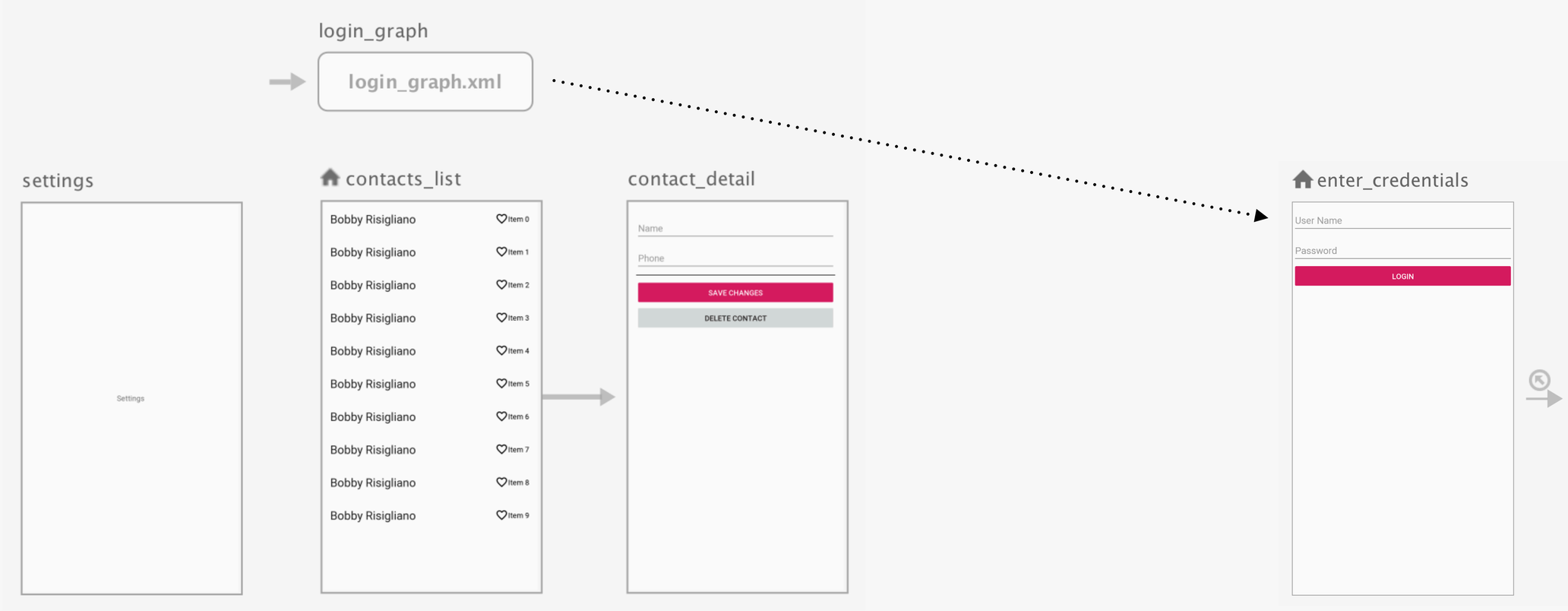
Conditional Starting Destination



First Time User Experience



First Time User Experiences



First Time User Experiences

```
<?xml version="1.0" encoding="utf-8"?>
<navigation android:id="@+id/contacts_graph"
            app:startDestination="@id/contacts_list" ...>

    <fragment android:id="@+id/contacts_list" ...>
    <fragment android:id="@+id/contact_detail" ...>
    <fragment android:id="@+id/settings" .../>
```

Nested Graph

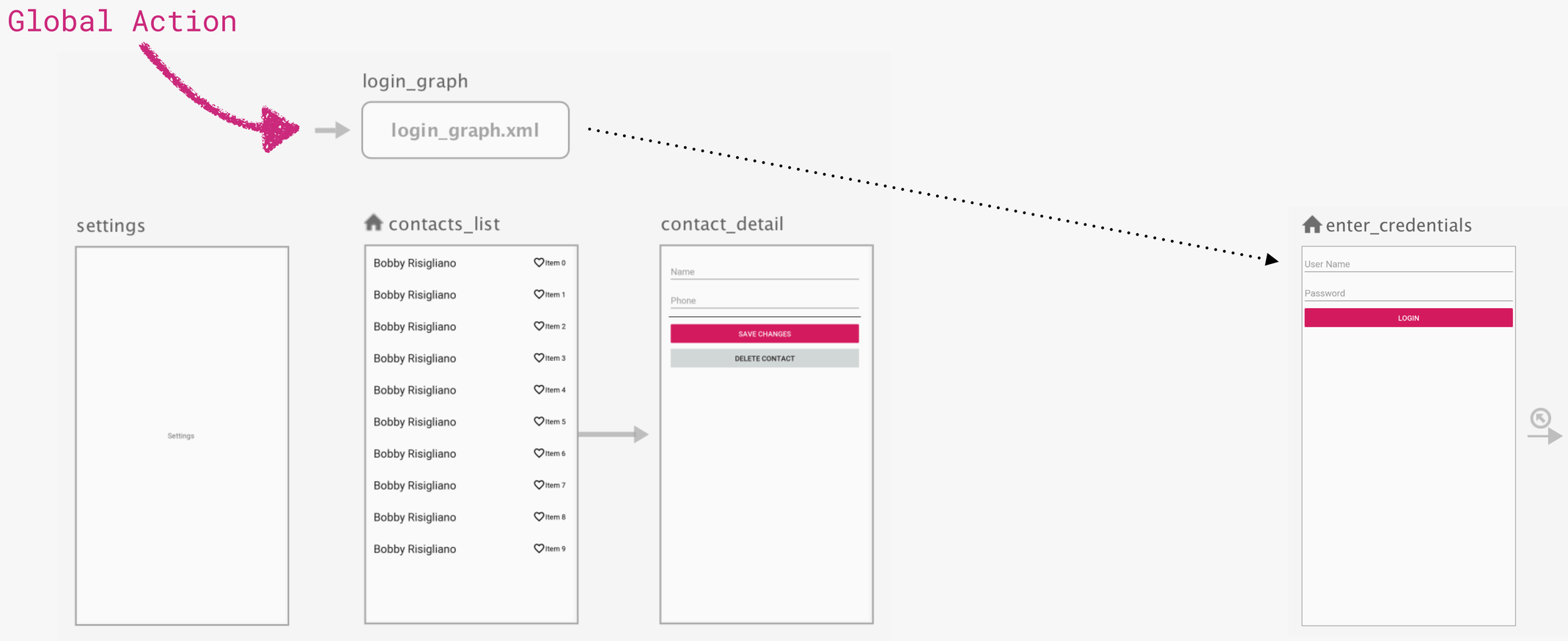
```
    <navigation android:id="@+id/login_graph"
                app:startDestination="@id/enter_credentials">
        <fragment
                android:id="@+id/enter_credentials"
                android:name="com.acme.contacts.security.EnterCredentialsFragment"
                android:label="Authentication Required"
                tools:layout="@layout/fragment_enter_credentials">
            <action
                    android:id="@+id/action_login_pop"
                    app:popUpTo="@id/login_graph"
                    app:popUpToInclusive="true" />
        </fragment>
    </navigation>

</navigation>
```

First Time User Experiences

```
<?xml version="1.0" encoding="utf-8"?>
<navigation android:id="@+id/contacts_graph"
            app:startDestination="@id/contacts_list" ...>

    <fragment android:id="@+id/contacts_list" ...>
    <fragment android:id="@+id/contact_detail" ...>
    <fragment android:id="@+id/settings" .../>

    <include android:id="@+id/login_graph"
             app:graph="@navigation/login_graph" />

</navigation>
```

Include as Separate Graph

First Time User Experiences



Main Navigation Graph

`contacts_graph.xml`

Login Graph

`login_graph.xml`

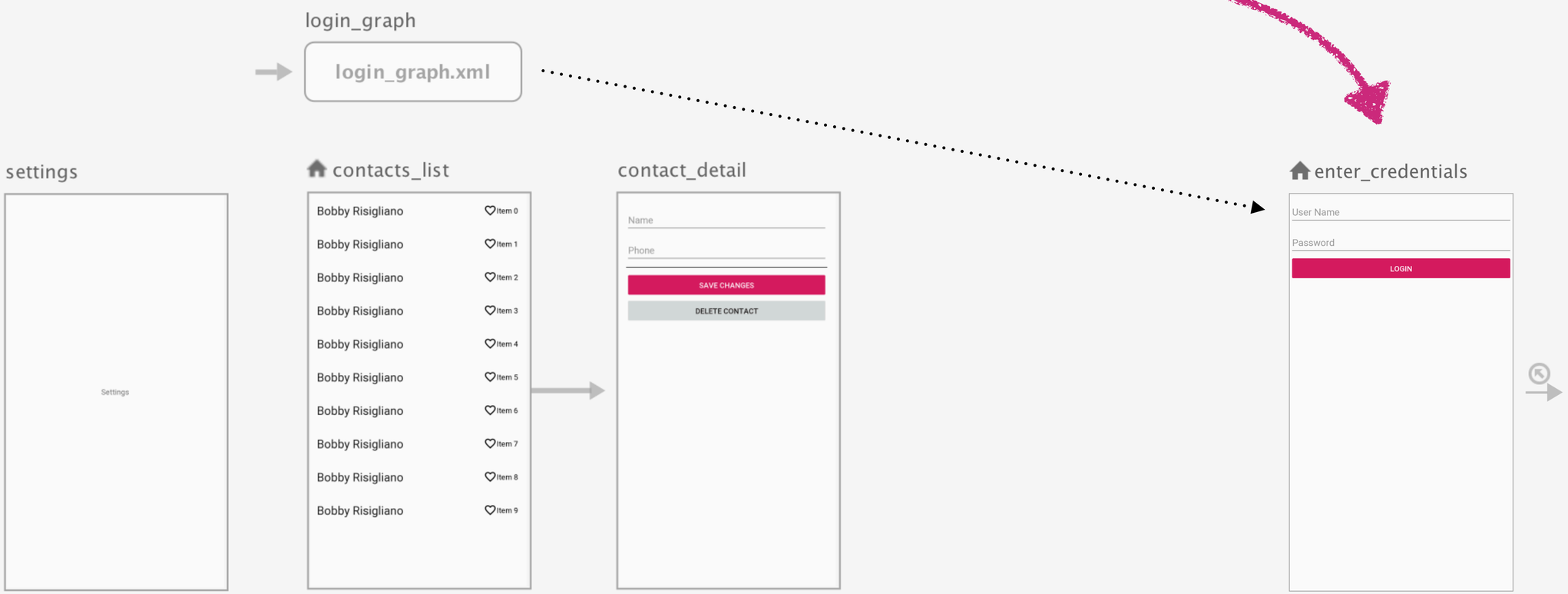
Global Action

```
<?xml version="1.0" encoding="utf-8"?>
<navigation android:id="@+id/contacts_graph"
            app:startDestination="@id/contacts_list" ...>

    <fragment android:id="@+id/contacts_list" ...>
    <fragment android:id="@+id/contact_detail" ...>
    <fragment android:id="@+id/settings" .../>

    <include android:id="@+id/login_graph"
             app:graph="@navigation/login_graph" />

    <action android:id="@+id/action_global_login"
            app:destination="@id/login_graph"
            app:launchSingleTop="true" />

</navigation>
```

Global Action

First Time User Experiences

1. If user is unauthenticated, redirect to login



2. Return to secured area once authenticated

Global Navigation

```
class ContactDetailFragment : SecureFragment() {

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

        super.onViewCreated(view, savedInstanceState)


    }
}
```

Global Navigation

```
abstract class SecureFragment : Fragment() {

    val authenticationViewModel by activityViewModels<AuthenticationViewModel>()

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

    private fun popBackStackOrExit() { ... }
}
```

Global Navigation

```
abstract class SecureFragment : Fragment() {

    val authenticationViewModel by activityViewModels<AuthenticationViewModel>()

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

        authenticationViewModel.authenticationStatus.observe(viewLifecycleOwner, Observer { authStatus ->
            when(authStatus) {
                UNAUTHENTICATED -> navController.navigate(actionGlobalLogin())
                USER_DECLINED -> popBackStackOrExit()
                else -> Unit
            }
        })
    }

    private fun popBackStackOrExit() { ... }
}
```

Global Navigation

```
abstract class SecureFragment : Fragment() {

    val authenticationViewModel by activityViewModels<AuthenticationViewModel>()

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

        authenticationViewModel.authenticationStatus.observe(viewLifecycleOwner, Observer { authStatus ->
            when(authStatus) {
                UNAUTHENTICATED -> navController.navigate(actionGlobalLogin())
                USER_DECLINED -> popBackStackOrExit()
                else -> Unit
            }
        })
    }

    private fun popBackStackOrExi
}
```

```
enum class AuthenticationStatus {
    UNAUTHENTICATED, // user needs to authenticate state
    AUTHENTICATED, // user is in authenticated state
    USER_DECLINED, // user has declined to authenticate
}
```

Global Navigation

```
abstract class SecureFragment : Fragment() {

    val authenticationViewModel by activityViewModels<AuthenticationViewModel>()


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

        authenticationViewModel.authenticationStatus.observe(viewLifecycleOwner, Observer { authStatus ->
            when(authStatus) {
                UNAUTHENTICATED -> navController.navigate(actionGlobalLogin())
                USER_DECLINED -> popBackStackOrExit()
                else -> Unit
            }
        })
    }

    private fun popBackStackOrExit() {
        if(!navController.popBackStack()) {
            requireActivity().finish()
        }
    }
}
```

Lab 3: Deep Linking + Conditional Navigation

Thanks!

Eric Maxwell

twitter @emmax

www.bignerdranch.com

Custom Destination Types