CMPS 312

Read Chapters 12, 13 & 15

Navigation

Dr. Abdelkarim Erradi

CSE@QU

Navigation

The act of moving between screens of an app to complete tasks

Designing effective navigation = Simplify the user journey

Outline

1. Communicating Between Activities
2. Menus and Toolbars
3. Navigation Component
4. Dialog Box

Communicating Between Activities

Back

Using Multiple Activities

- How do we **navigate** to a new screen?
  - Start a new Activity using an **Intent**

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

- What is an **Intent**?
  - Enables communication between Activities
  - It is a **messaging object** to communicate to the system that some action should be carried out
  - Implicit vs Explicit Intents: choosing a generic action vs specifying a specific app component

Implicit vs Explicit Intents

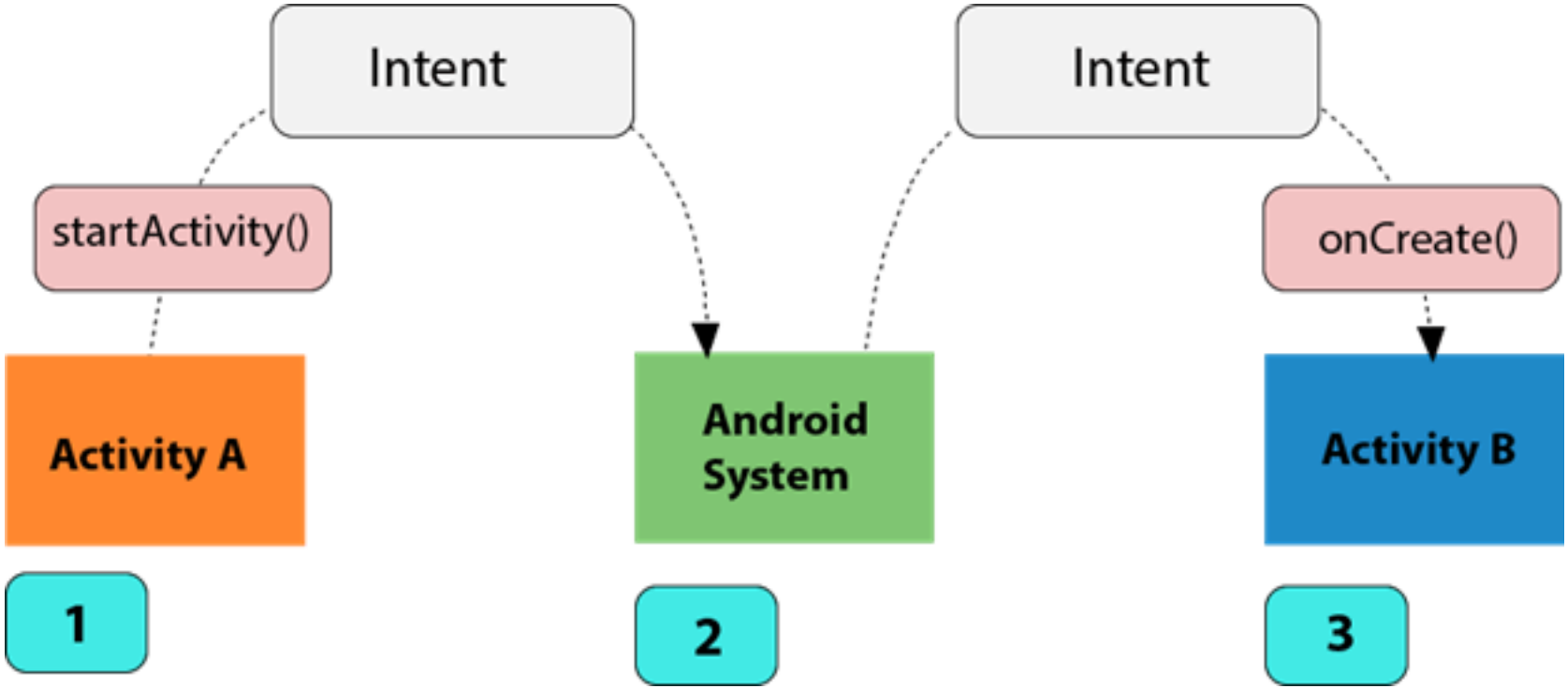
- **Explicit** intents can be used to start a specific Activity
- Implicit intents describe a general action such as display contacts, broadcast a message, dial a phone call etc.
  - **Display contact**: `ACTION_VIEW -> content://contacts/people/1`
  - **Dial a number**: `ACTION_DIAL -> content://contacts/people/1`
  - **Send an email**: `ACTION_SEND -> EXTRA_EMAIL, EXTRA_SUBJECT`
  - Specifies an **ACTION** and **DATA** (parameters expected by the action)
  - Implicit intents can be handled by **a component in the system** registered to handle that intent type

Explicit Intent

- Explicit intents can be used to start a specific Activity

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

startActivity(intent)
```



Implicit Intent

- **Implicit Intent** does not specify the component name. Another app will handle it

```
intent= Intent(Intent.ACTION_VIEW, Uri.parse(
"https://www.qu.edu.qa/"))
startActivity(intent)
```

Passing Data with Intents

- Pass data

```
val intent = Intent(this, RegisterActivity::class.java)
// Pass student ID and student name with Intent so it can be
// used by RegisterActivity when it’s started
intent.putExtra("id", 235789)
intent.putExtra("name", "Peter Pan")
startActivity(intent)
```

- Get passed data

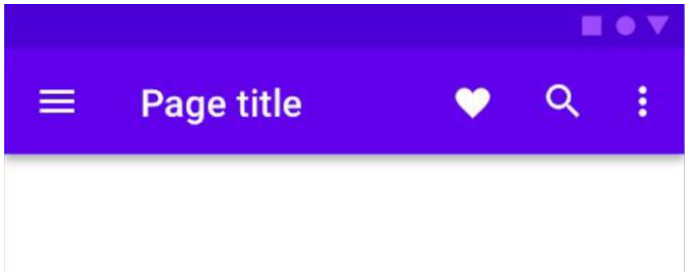
```
override fun onCreate(savedInstanceState: Bundle?) {
     ...
    // Read data sent by the caller
    val id = intent.getIntExtra("id", 0)
    val name = intent.getStringExtra("name")
}
```

Menus and Toolbars

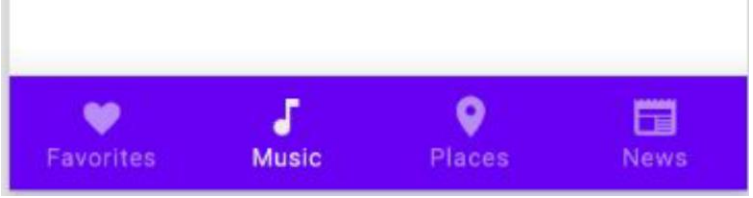
Back

Menus and Toolbars

- **AppBar**
  - Info and actions related to the current screen
  - Typically positioned on top and has Title, Menu items, Drawer button, Back button



- **Bottom Navigation**
  - Provides movement between **top-level destinations** in an app
  - Positioned on the Bottom of screen and typically has Label/Icon and Notification badges, Selection / Reselection



Navigation drawer

Floating Action Button

Selection Criteria

Component	Use for	# destinations
Navigation drawer	Top-level destinations	5+
Bottom navigation bar	Top-level destinations	2-5
Tabs	Any level of hierarchy	2+

Navigation Component

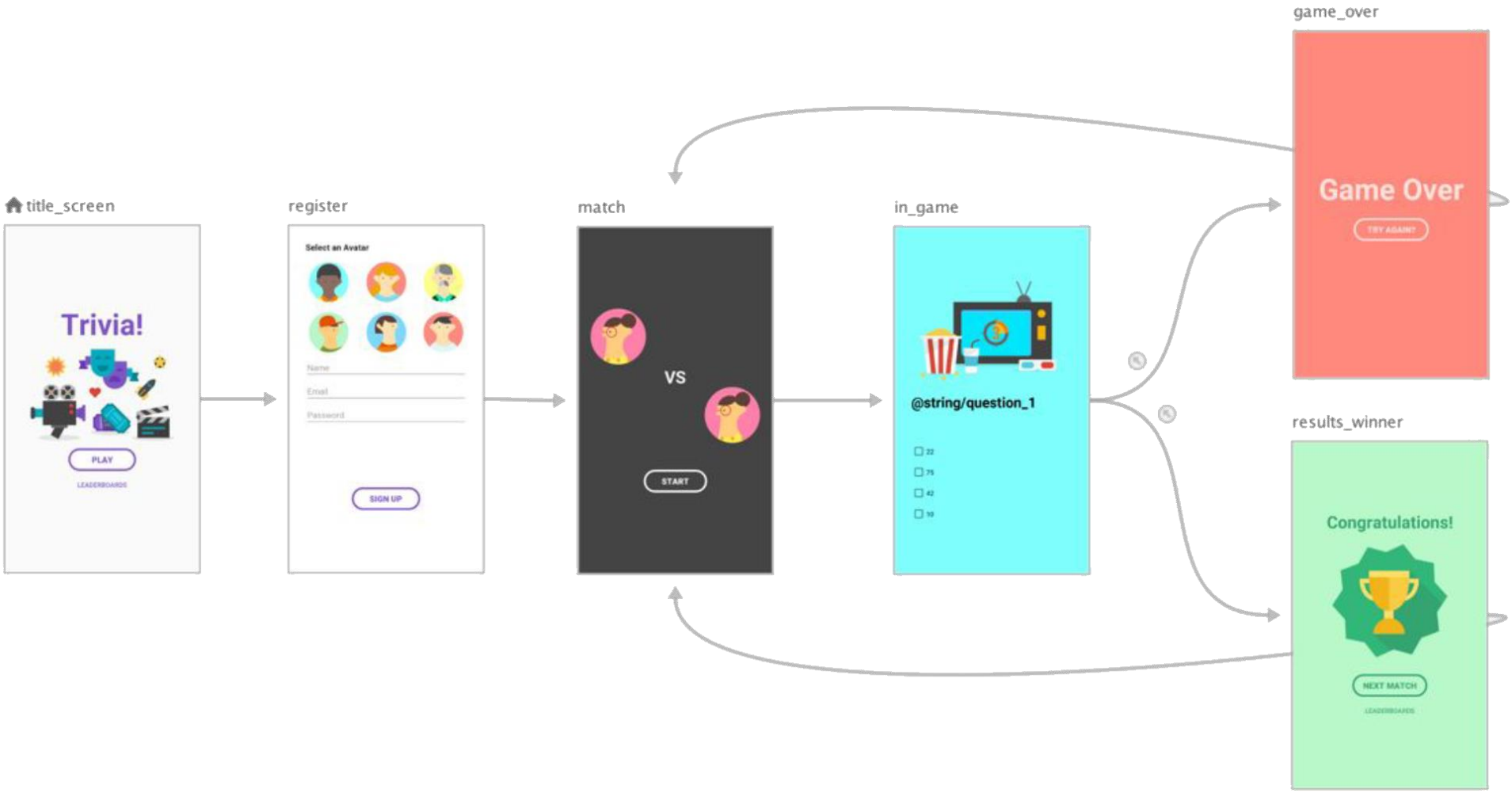
A framework for navigating between ‘destinations’ within an app

Back

Navigation Component

- Helps manage fragment transitions
- Integration With Material Design UI (e.g., auto setting toolbar title)
- Visual representation of app navigation => **Navigation Graph** + Navigation design editor
- Graph defines Destinations & Actions that trigger transition to the next destination
- Compile-time validation of destination transactions
- Compile-time validation of fragment arguments

Example Navigation Graph



Key Components

Navigation Graph	NavHost	NavController
XML representation of app navigation	A container where fragments will be displayed	Manages the transitions between graph destinations

Dependencies

```
// Project/build.gradle
def nav_version = "2.3.0"
classpath "androidx.navigation:navigation-safe-args-gradle-plugin:$nav_version"
```

```
// Module:app/build.gradle
def nav_version = "2.3.0"
implementation "androidx.navigation:navigation-fragment-ktx:$nav_version"
implementation "androidx.navigation:navigation-ui-ktx:$nav_version"
```

```
// Module:app/build.gradle
apply plugin: "androidx.navigation.safeargs.kotlin"
```

Implementing Navigation

Create A Nav Graph

- Create an XML file to define the app's navigation graph

Add A Start Destination

- Define which Fragment to show when app starts

Add Nav Graph to NavHost

- Connect the XML nav graph to the container that will display Fragments

Navigate to Destinations Using NavController

- NavController will manage the Fragment transitions and UI updates

Resources

- Get started with the Navigation component
  - https://developer.android.com/guide/navigation/navigation-getting-started

- Navigation Component codelab
  - https://codelabs.developers.google.com/codelabs/kotlin-android-training-add-navigation/