# Mobile Application Development

Produced by

David Drohan (ddrohan@wit.ie)

Department of Computing & Mathematics
Waterford Institute of Technology
http://www.wit.ie

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

# AndroidX Navigation Component

# Agenda

- Overview
- Navigation Component Key Parts
- Component Benefits
- Navigation Drawers

# Overview – AndroidX & Jetpack

- **AndroidX** is the open-source project that the Android team uses to develop, test, package, version and release libraries within **Jetpack**
- **Jetpack** is a suite of libraries, tools, and guidance to help developers write high-quality apps easier. These components help you follow best practices, free you from writing boilerplate code, and simplify complex tasks, so you can focus on the code you care about

- **Navigation** is just one small component of these libraries!

# Overview –

# AndroidX & Jetpack

## Foundation

Foundation components provide cross-cutting functionality like backwards compatibility, testing and Kotlin language support.

### Android KTX

Write more concise, idiomatic Kotlin code

### AppCompat

Degrade gracefully on older versions of Android

### Auto

Components to help develop apps for Android Auto

### Benchmark

Quickly benchmark your Kotlin-based or Java-based code from within Android Studio

### Multidex

Provide support for apps with multiple DEX files

### Security

Read and write encrypted files and shared preferences by following security best practices.

### Test

An Android testing framework for unit and runtime UI tests

### TV

Components to help develop apps for Android TV

### Wear OS by Google

Components to help develop apps for Wear

## Architecture

Architecture components help you design robust, testable and maintainable apps.

### Data Binding

Declaratively bind observable data to UI elements

### Lifecycles

Manage your activity and fragment lifecycles

### LiveData

Notify views when underlying database changes

### Navigation

Handle everything needed for in-app navigation

### Paging

Gradually load information on demand from your data source

### Room

Fluent SQLite database access

### ViewModel

Manage UI-related data in a lifecycle-conscious way

### WorkManager

Manage your Android background jobs

## Behavior

Behavior components help your app integrate with standard Android services like notifications, permissions, sharing and the Assistant.

### CameraX

Easily add camera capabilities to your apps

### Download manager

Schedule and manage large downloads

### Media & playback

Backwards-compatible APIs for media playback and routing (including Google Cast)

### Notifications

Provides a backwards-compatible notification API with support for Wear and Auto

### Permissions

Compatibility APIs for checking and requesting app permissions

### Preferences

Create interactive settings screens

### Sharing

Provides a share action suitable for an app's action bar

### Slices

Create flexible UI elements that can display app data outside the app

## UI

UI components provide widgets and helpers to make your app not only easy, but delightful to use. Learn about Jetpack Compose, which helps simplify UI development.

### Animation & transitions

Move widgets and transition between screens

### Emoji

Enable an up-to-date emoji font on older platforms

### Fragment

A basic unit of composable UI

### Layout

Lay out widgets using different algorithms

### Palette

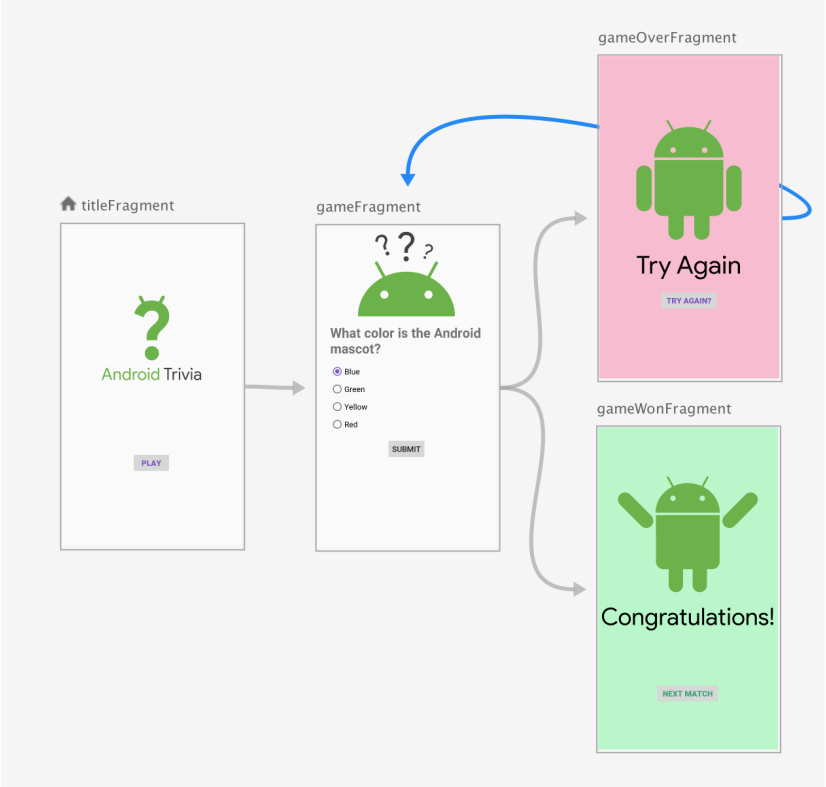
Pull useful information out of color palettes

# Overview – Navigation

- **Navigation** refers to the interactions that allow users to navigate across, into, and back out from the different pieces of content within your app
- **Android Jetpack's Navigation component** helps you implement navigation, from simple button clicks to more complex patterns, such as app bars and the navigation drawer
- The Navigation component also ensures a consistent and predictable user experience by adhering to an established set of principles

# Navigation Component Key Parts

- **Navigation graph**: An XML resource that contains all navigation-related information in one centralized location.
- This includes all of the individual content areas within your app, called destinations, as well as the possible paths that a user can take through your app.



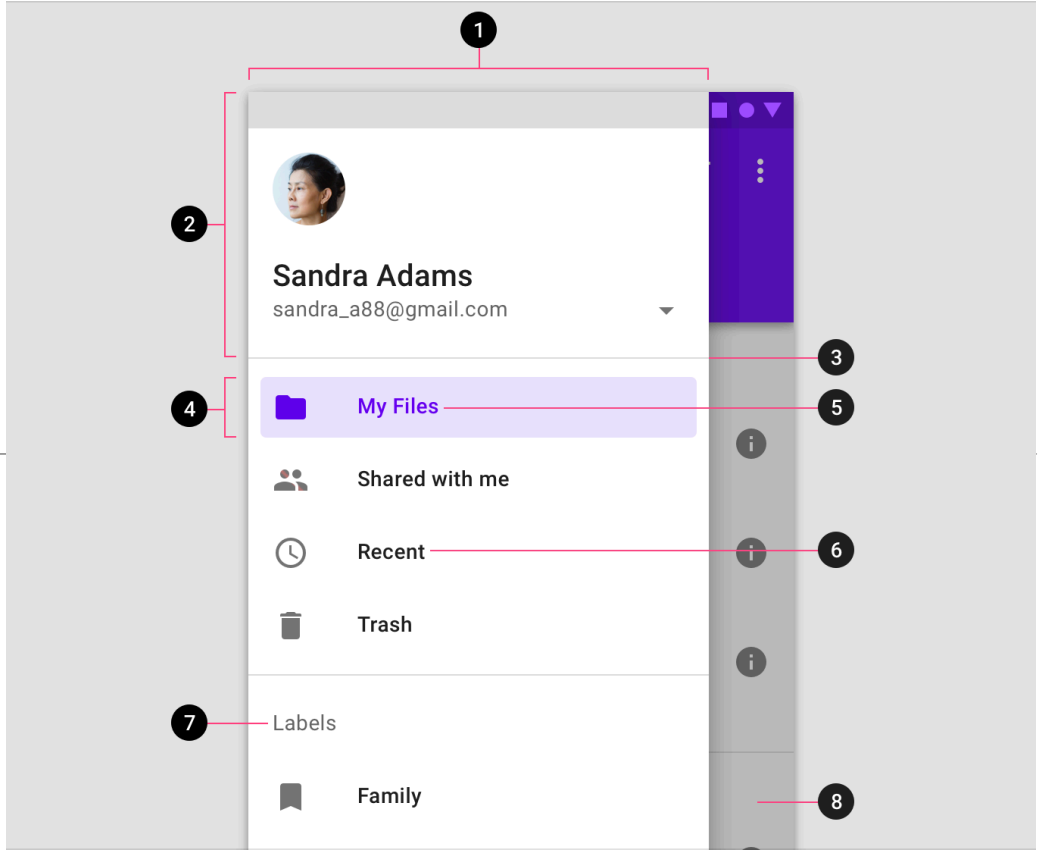
# Navigation Component Key Parts

- **NavHost**: An empty container that displays destinations from your navigation graph. The Navigation component contains a default **NavHost** implementation, **`NavHostFragment`**, that displays fragment destinations.
- **NavController**: An object that manages app navigation within a **NavHost**. The **NavController** orchestrates the swapping of destination content in the **NavHost** as users move throughout your app.

# Navigation Component Benefits

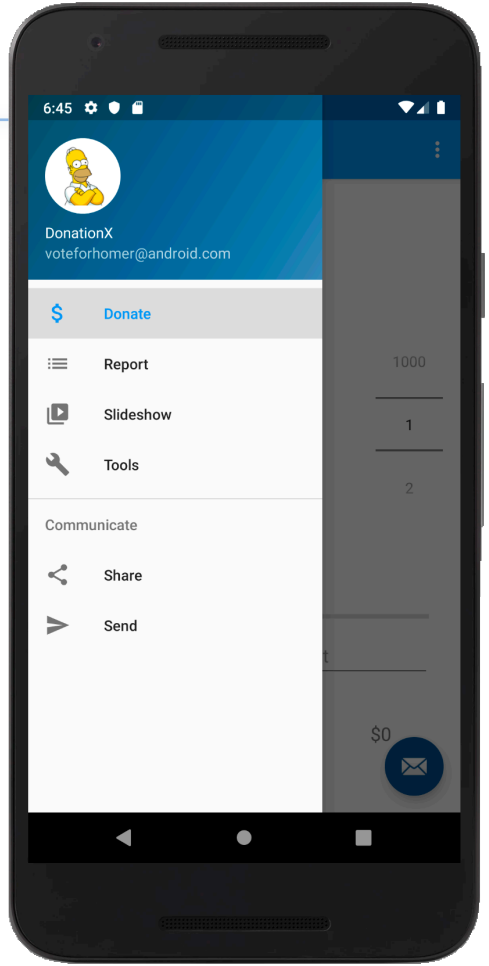
- Handling fragment transactions
- Handling Up and Back actions correctly by default
- Providing standardized resources for animations & transitions
- Implementing and handling deep linking
- Including Navigation UI patterns, such as **navigation drawers** and bottom navigation, with minimal additional work
- **ViewModel** support - you can scope a **`ViewModel`** to a navigation graph to share UI-related data between the graph's destinations

# Navigation Drawer Overview



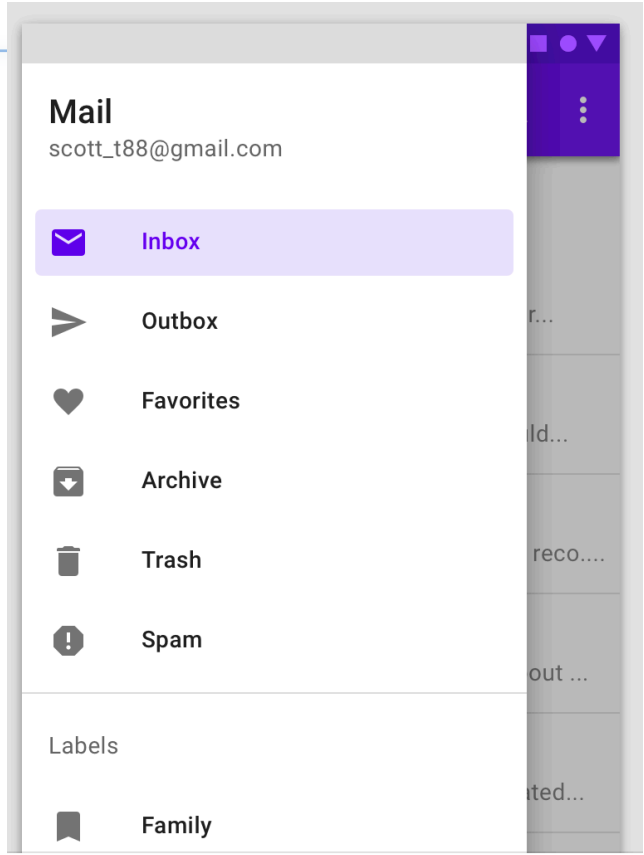
# Navigation Drawer Overview

- https://developer.android.com/training/implementing-navigation/nav-drawer.html
- The navigation drawer is a panel that displays the app's main navigation options on the left edge of the screen. It is hidden most of the time, but is revealed when the user swipes a finger from the left edge of the screen or, while at the top level of the app, the user touches the app icon in the action bar

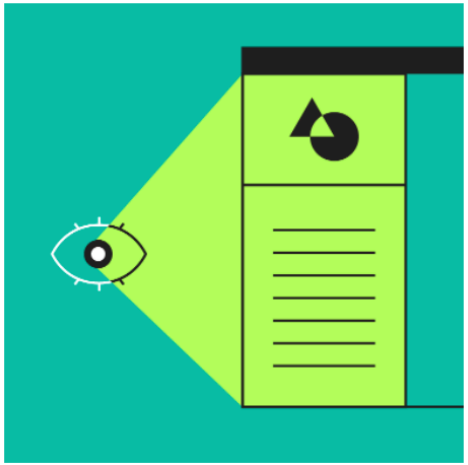


# Navigation Drawer Overview

- Navigation drawers are recommended for:
  - Apps with five or more top-level destinations *
  - Apps with two or more levels of navigation hierarchy
  - Quick navigation between unrelated destinations

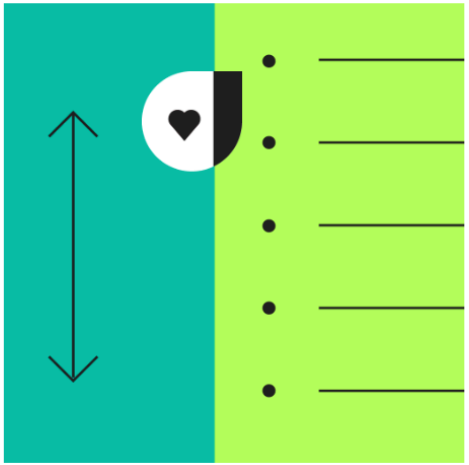


# Navigation Drawer Principles



### Identifiable

The placement and list-style content of navigation drawers clearly identify them as navigation.



### Organized

Navigation drawers order destinations according to user importance, with frequent destinations first and related ones grouped together.

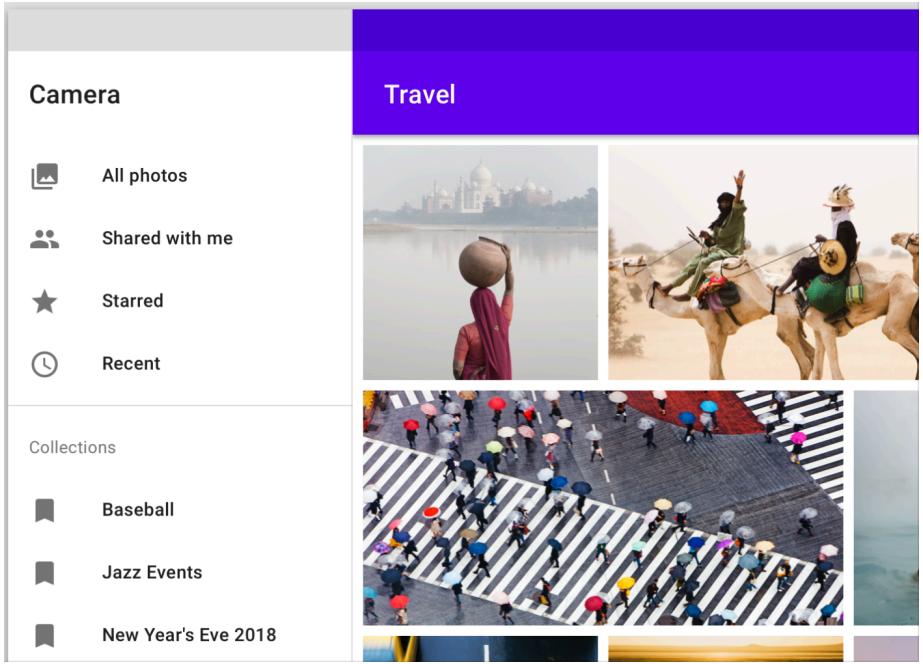
### Contextual

Navigation drawers can be shown or hidden to accommodate different app layouts.

# Nav Drawer Types

## Standard

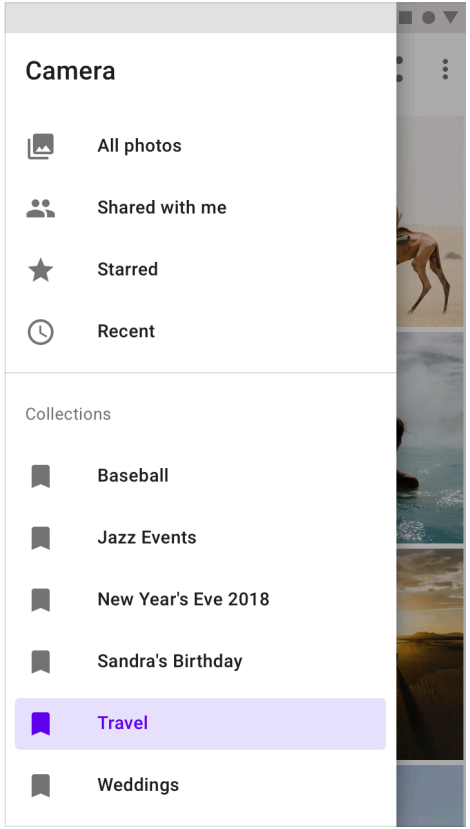
- allow users to simultaneously access drawer destinations and app content. They are often co-planar with app content and affect the screen's layout grid
- can be permanently visible or opened and closed by tapping a navigation menu icon. They can be used on tablet and desktop only. On mobile, modal drawers are used instead



# Nav Drawer Types

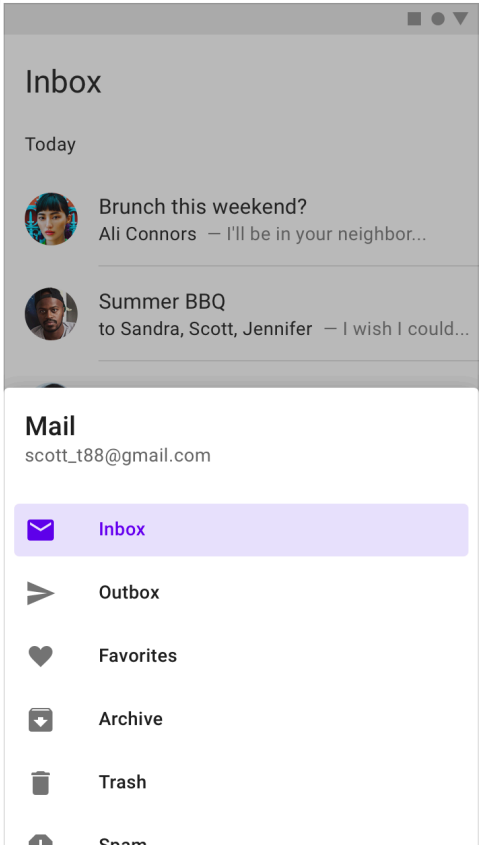
## Modal

- use a scrim to block interaction with the rest of an app's content. They are elevated above most app elements and don't affect the screen's layout grid
- primarily for use on **mobile**, where screen space is limited. They can be replaced by standard drawers on tablet and desktop



# Nav Drawer Types

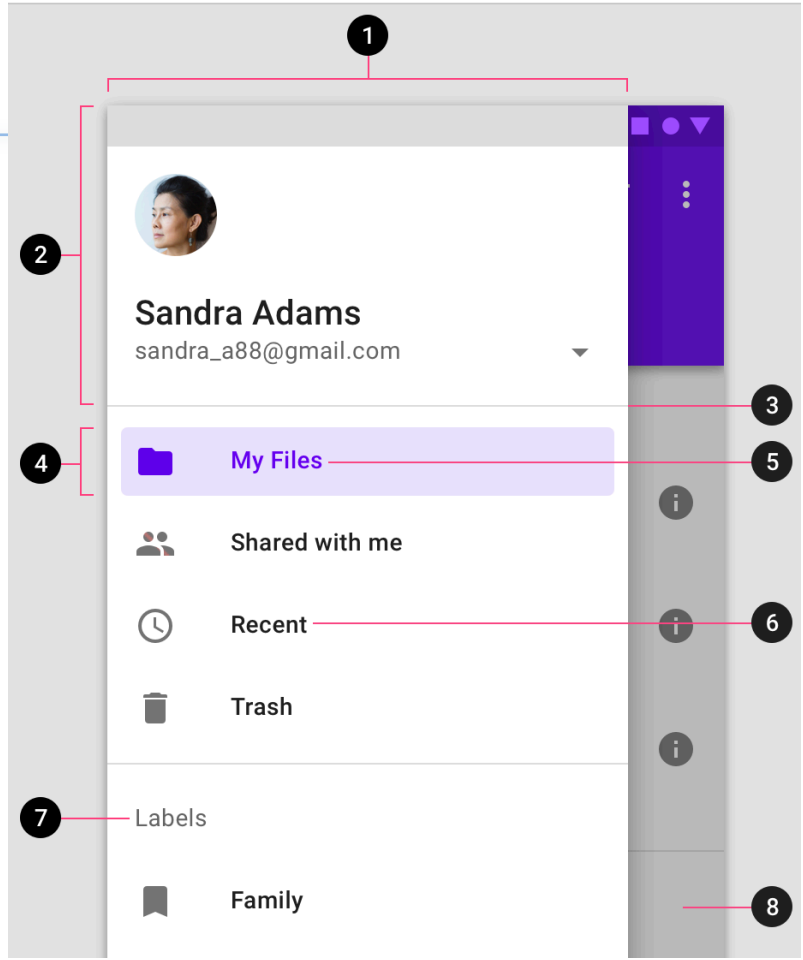
- Bottom
  - are a specialized type of modal drawer for use with a bottom app bar
  - for increased reachability from the bottom app bar's menu icon, they open from the bottom of the screen rather than the side



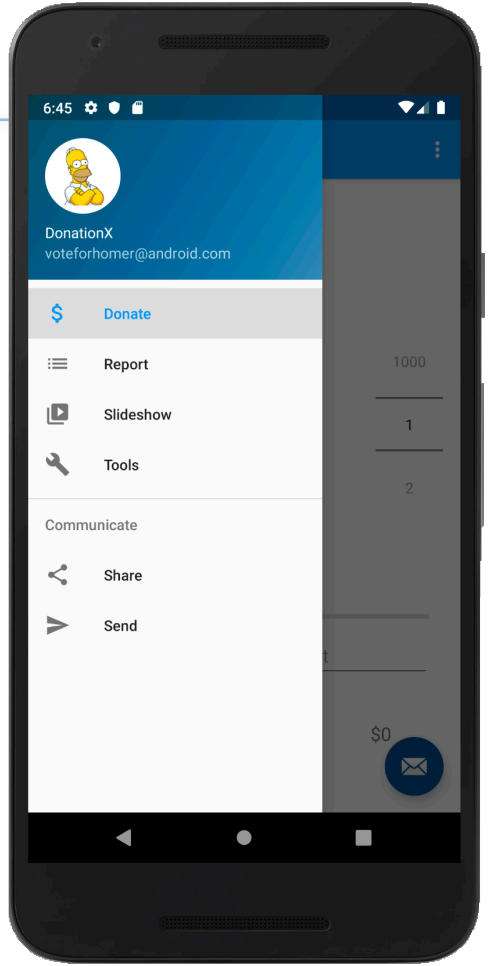
# Nav Drawer Anatomy

- contains a list embedded within a sheet
- can be enhanced with headers and dividers to organize longer lists

1. Container
2. Header (optional)
3. Divider (optional)
4. Active text overlay
5. Active text
6. Inactive text
7. Subtitle
8. Scrim (modal only)



# Navigation Drawer
# &
# Fragments
# in
# Donation




# References

Sources:

https://blog.mindorks.com/android-navigation-drawer-in-kotlin
https://medium.com/@umang.burman.micro/navigation-drawer-with-navigation-component-4f032bfdeae6
https://developer.android.com/guide/navigation
https://medium.com/@myric.september/navigation-in-android-navigation-architecture-component-a1f103a6cc52
https://material.io/components/navigation-drawer/#


Thanks.