Navigation.

Navigation Library is a library to handle all the navigation needs of an app. The word "navigation " in this context,

it would mean all the paths a user could take on pressing certain buttons or links. There are 3 components involved:

1. Navigation graph : a `anyname.xml` file present in res/navigation that holds a complete graph of the various paths a user could go to. you would define the complete details about fragments and activitites and how they are connected to each other here. These fragments/activites are called **Destinations.**
2. Navhost:

```

The Navigation component consists of three key parts that are described below:

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.

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.

```

Principles of navigation

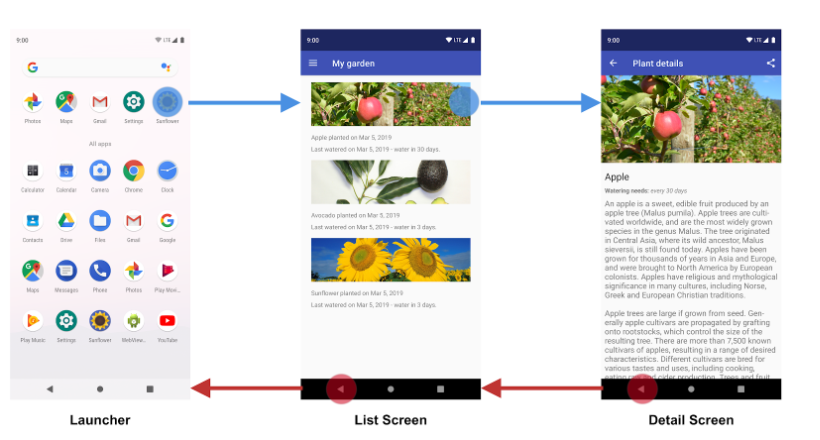
There are some certain guidelines that the google devs recommends, which should be followed by any app, weather implementing an navigation library or not. these are:

1. **Fixed start destination:**

There should be a single point of start and end for an app. No matter wherever user navigates off to, they shouldn’t be directly exiting, without going back to the “Home Screen”.

The user could wander off to multiple screens, but when they presses the back button, each back button would point back to last caller component, and ultimately reaching the original starting activity, thereby making the starting activity as both the start and end point of an app.

Thus if user went from activity launcher->A->B->C->D, they should be propagating back similarly, i.e D->C(on back/up press)->B(on back/upPress )->A->exit(on back press only)

****

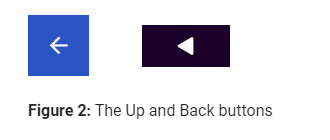
Although certain clauses exists for non startend points like:

* **Introduction Slider Activity ( which is shown only once to user on app installed)** : Not considered as start-end screens , but rather a one-time set of screens. the navigationlib handles it using **conditionals(covered later)**
* **splash screen( which is shown for a few seconds before opening main screen)** : <Not sure>
* **Deep links activity( which are activites in an that automatically open when clicked in a google search)** : considered part of the back propagation such that every applink activity must have a back button which should point to some intermediate activity or start-end activity, such that user sees the start-end component before closing the app

1. **Roles of the back button and up button.**

both the back button and the up button plays the **role of moving down in the backstack,** but only the back button should have **the power to clear backstack and exit .**

**Note:** This rule also applies to the deep links. thus when th user in deep link activity presses up button, they go back into the app, but when the presses the back button, they directly exit

****

1. **Deep linking and synthetic backstack :**  deep links replae whatever backstack on the screen with a synthetica back stack. for eg, if user is currently at A->B->C->D, and then presses home, searches google and click app’s deeplink , the google search app will open the deep link activity K with a new backstack as A->K

**Adding navigation components:**

**app/build.gradle**android **{**

**//...** compileOptions **{** sourceCompatibility JavaVersion.***VERSION\_1\_8*** targetCompatibility JavaVersion.***VERSION\_1\_8* }  
}**dependencies **{  
  
 def** nav\_version = **"2.3.0"***// note: must have java1.8  
  
 // Java language implementation.  
 //*implementation **"androidx.navigation:navigation-fragment:**$nav\_version**"  
 //**implementation **"androidx.navigation:navigation-ui:**$nav\_version**"** *//safe args plugin ( for passing data between fragments): in projectroot/gradle  
  
 // Dynamic Feature Module Support :Use if your app has a modular architecture* implementation **"androidx.navigation:navigation-dynamic-features-fragment:**$nav\_version**"** *// Kotlin* implementation **"androidx.navigation:navigation-fragment-ktx:**$nav\_version**"** implementation **"androidx.navigation:navigation-ui-ktx:**$nav\_version**"***// must have ktx enabled  
   
 // testing modules(****todo)*}**

**project-root/build.gradle***// Top-level build file where you can add configuration options common to all sub-projects/modules.*buildscript **{** ...dependencies **{** classpath **"com.android.tools.build:gradle:4.2.0-alpha07"** *// NOTE: Do not place your application dependencies here; they belong  
 // in the individual module build.gradle files* **def** nav\_version = **"2.3.0"** classpath **"androidx.navigation:navigation-safe-args-gradle-plugin:**$nav\_version**"  
  
 }  
}  
...**

**1. creating a navigation graph**

a) Create a navigation directory. add a navigation file in it.