20CSI605 - Mobile Application Development

Responsive layout

Responsive layout

- Flutter, being a cross-platform app development framework, supports devices with widely varying screen sizes:
- ✔ It can run on a device as small as a smartwatch to devices like a large TV.
- ✔ It's always a challenge to adapt your app to such a variety of screen sizes and pixel densities using the same codebase.

Responsive layout

Android approach

- In order to handle different screen sizes and pixel densities, the following concepts are used in Android

1. ConstraintLayout

- ✔ One of the revolutionary tools introduced in the Android world for UI design is the ConstraintLayout.
- ✔ It can be used for creating flexible and responsive UI designs that adapt to different screen sizes and dimensions.
- ✔ ConstraintLayout allows you to specify the position and size for each view according to spatial relationships with other views in the layout.

Responsive layout

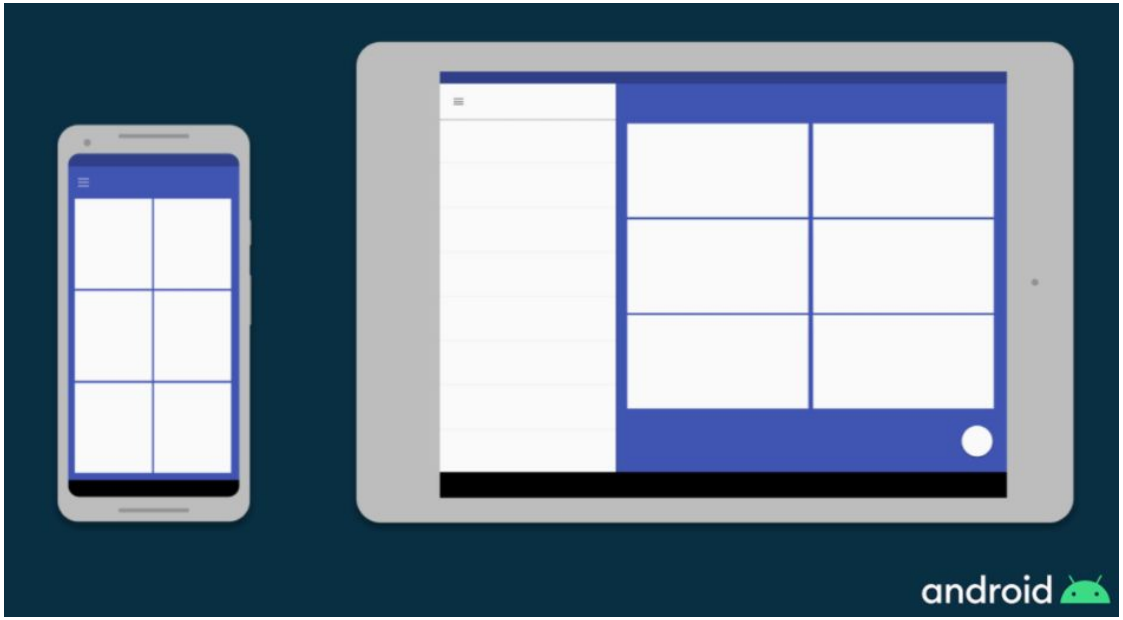
1. ConstraintLayout

- This can be applied to devices like smartwatches, which have very little screen real estate, and resizing the components to fit that screen size might result in a weird UI.

Responsive layout

2. Alternative layouts

- To solve the above issue, you can use alternative layouts for different-sized devices.
- For example, you can use **split view** in devices like tablets to provide a good user experience and use the large screen real estate wisely.



Responsive layout

2. Alternative layouts

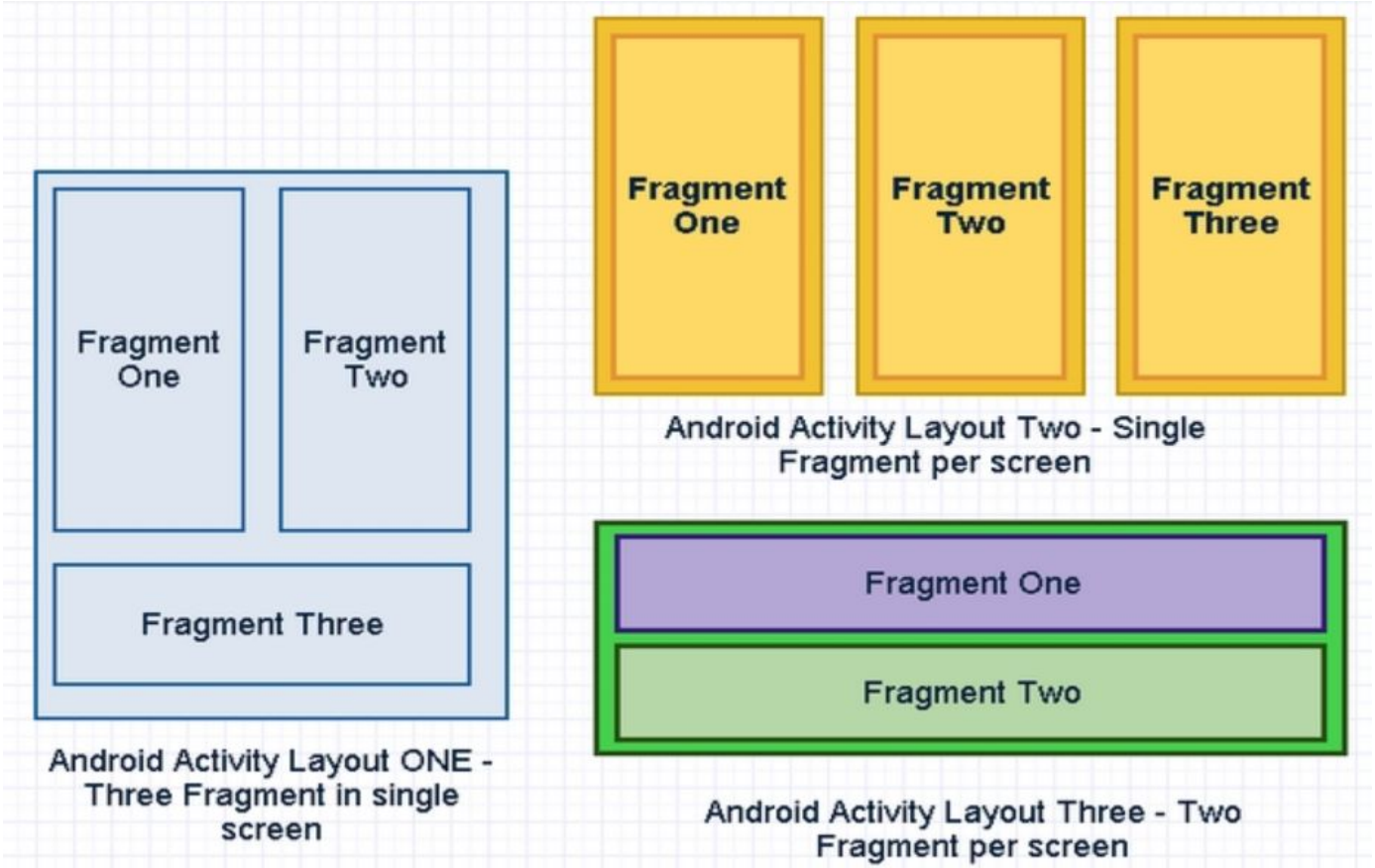
- In Android, you can define **separate layout files** for different screen sizes, and the Android framework handles the switching between these layouts automatically as per the screen size of the device.

3. Fragments

- Using Fragment, you can extract your UI logic into separate components so that while designing multi-pane layouts for large screen sizes, you do not have to define the logic separately.
- You can just reuse the logic that you have defined for each fragment

Responsive layout

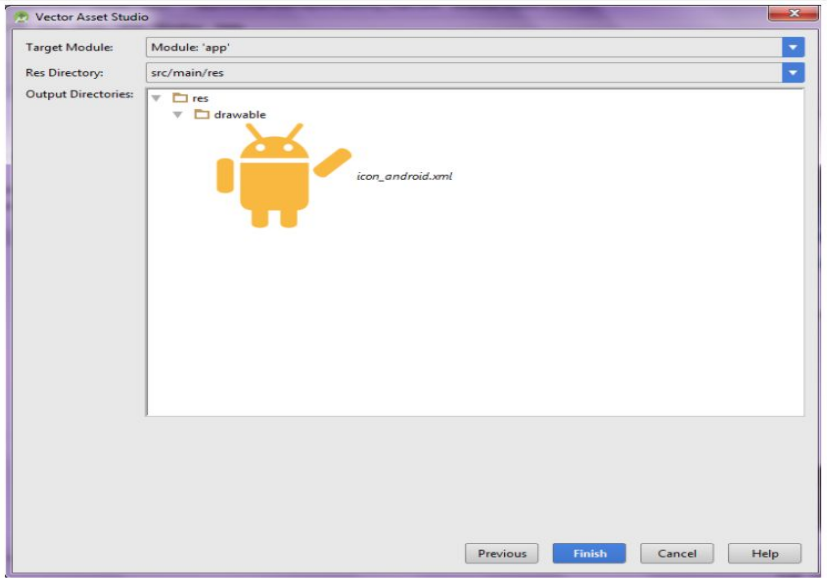
3. Fragments



Responsive layout

4. Vector graphics

- As opposed to being created using pixel bitmaps, vector graphics are images defined in XML files as paths and colors.
- They can scale to any size without scaling artifacts.
- In Android, you can use VectorDrawable for any kind of illustration, such as icons.



Responsive layout using Flutter

A responsive grid layout can be said to adapt to different screen size, and orientation, ensuring consistency across layouts**.**

Responsive grid properties include:

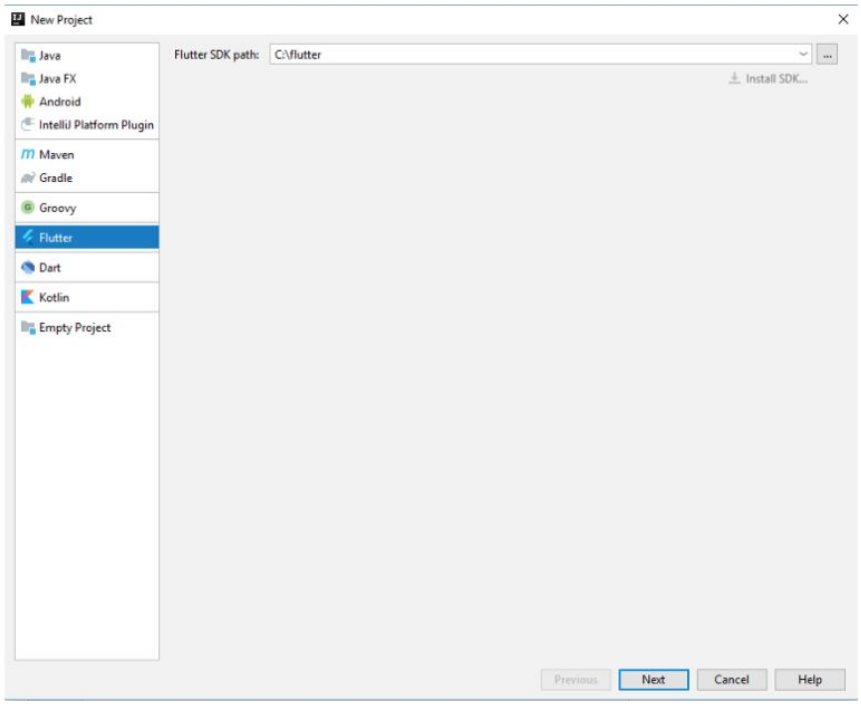
- **xs** stands for **extra small** (for phones — screens less than 768px wide)
- **sm** stands for **small** (for tablets — screens equal to or greater than 768px wide)
- **md** stands for **medium** (for small laptops — screens equal to or greater than 992px wide)
- **lg** stands for **large** (for laptops and desktops — screens equal to or greater than 1200px wide)

Responsive layout using Flutter

Responsive grid properties include:

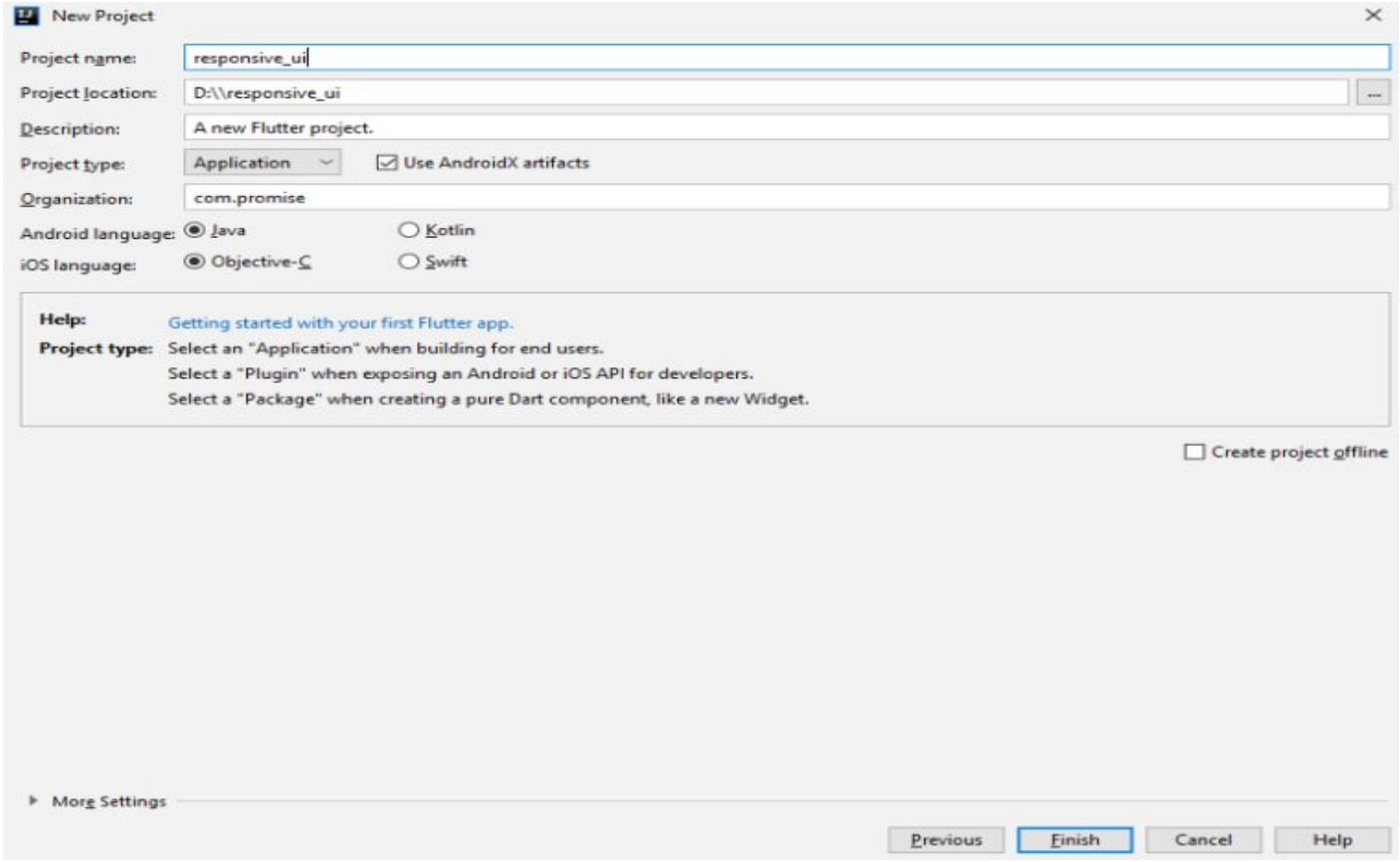
Responsive layout using Flutter

- Create a new flutter project using Visual Studio, IntelliJ or Android studio.
- If you have IntelliJ installed on your computer, click: File>>New>>Project, Select Flutter and click “Next”



Responsive layout using Flutter

Enter the project name and click “Finish”



Responsive layout using Flutter

Open “pubspec.yaml”, and install a responsive grid package

```
dependencies:
  responsive_grid: ^1.0.1
```

We can take the same approach using Android styles. First, setup a base style.

res/values/styles.xml

```
<style name="Container">
    <item name="android:layout_margin">0dp</item>
    <item name="android:padding">16dp</item>
    <item name="android:layout_width">match_parent</item>
    <item name="android:layout_height">match_parent</item>
    <item name="android:orientation">vertical</item>
    <item name="android:background">@drawable/container_background</item>
</style>
```

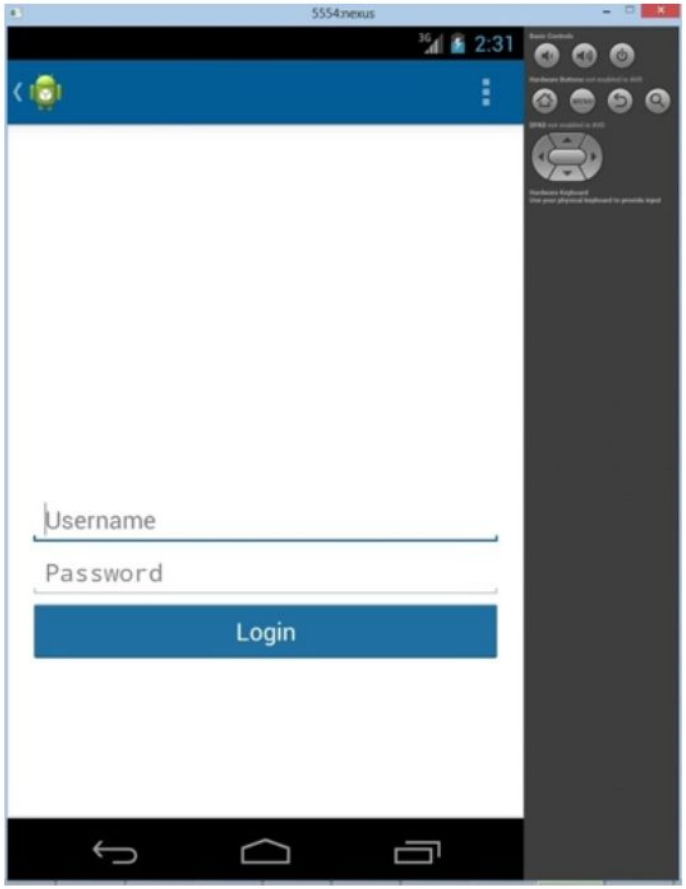
Responsive layout using Flutter

For tablets in portrait orientation, we add a bit more padding since the screen is larger.

res/values-sw600dp/styles.xml

```
<style name="Container">
    <item name="android:layout_margin">0dp</item>
    <item name="android:padding">32dp</item>
    <item name="android:layout_width">match_parent</item>
    <item name="android:layout_height">match_parent</item>
    <item name="android:orientation">vertical</item>
    <item name="android:background">@drawable/container_background</item>
</style>
```

Responsive layout using Flutter



THANK YOU