The University Of Sheffield.

COM4510/6510
Software Development for Mobile Devices

Lecture 3: Lifecycle and Layouts (Part 1)

Dr Po Yang
The University of Sheffield
po.yang@sheffield.ac.uk

Lecture Overview

- Part 1:
  - Activity's Lifecycle
  - Designing an app with Material Design
- Part 2:
  - Design Patterns
  - Styles and Themes
  - Layout Grids
- Lab tutorial :
  - Designing sensible layouts

The University Of Sheffield.

An Activity's Lifecycle

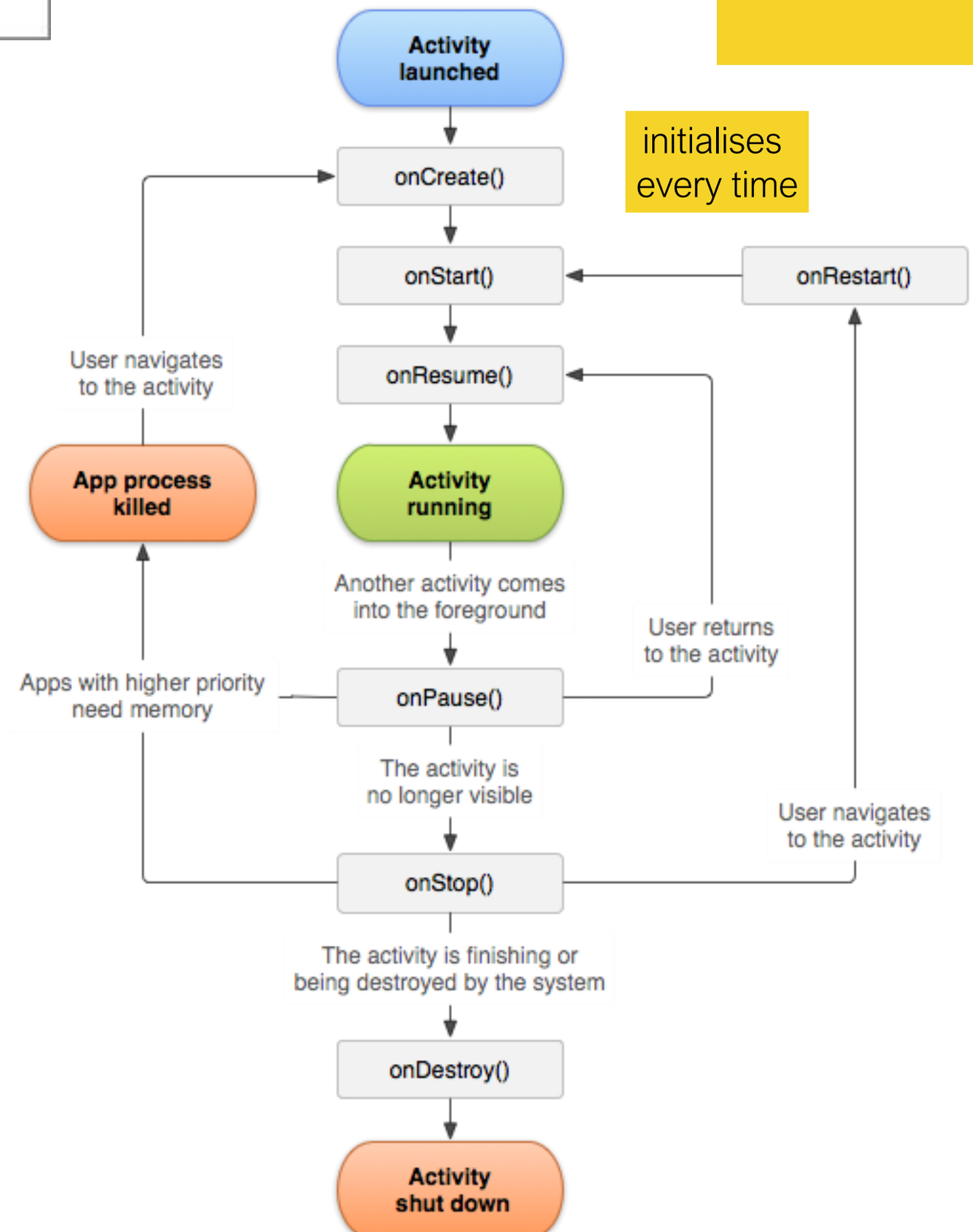
Lifecycle

- Any activity will go through **a cycle** of being opened and becoming visible to a phase **where it is not visible** (e.g. because you open another component)
- There are **different phases**:
  - Activity is created
  - Activity becomes **visible** to the user
  - Activity becomes **invisible** to the user (e.g. minimised)
  - Activity is **destroyed** (e.g. swiped out)
- The activity can move seamlessly through these phases and not necessarily in that order

The University Of Sheffield.

Initialises variable that will last for the lifetime of the activity (just once)

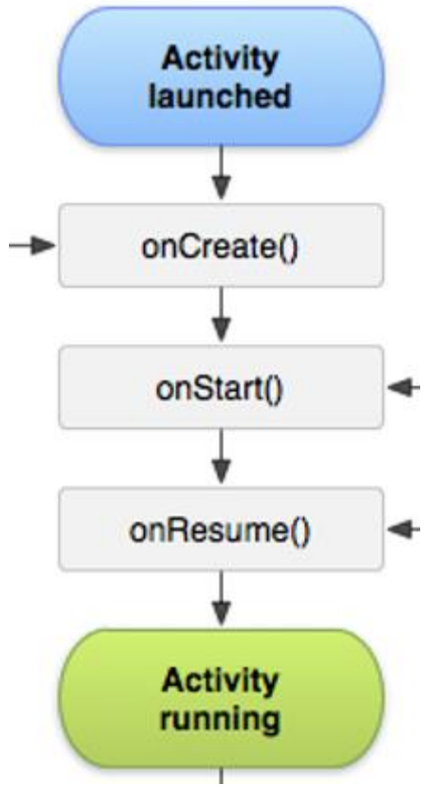
initialises every time



https://developer.android.com/reference/android/app/Activity.html#ActivityLifecycle

OnCreate

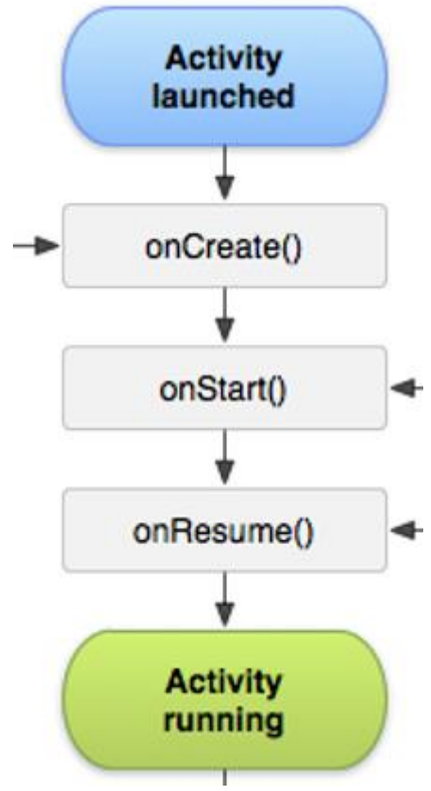
- Callbacks.
- **onCreate()**
  - You **must implement this callback**, which fires when the system creates your activity. Your implementation should initialize the essential components of your activity
    - You **must call setContentView()** to define the layout for the activity's user interface.
    - Your app should create any views programmatically or and bind data to containers such as lists and grids
- When onCreate() finishes, the next callback is always **onStart()**.



OnStart and onResume

You are not required to implement them

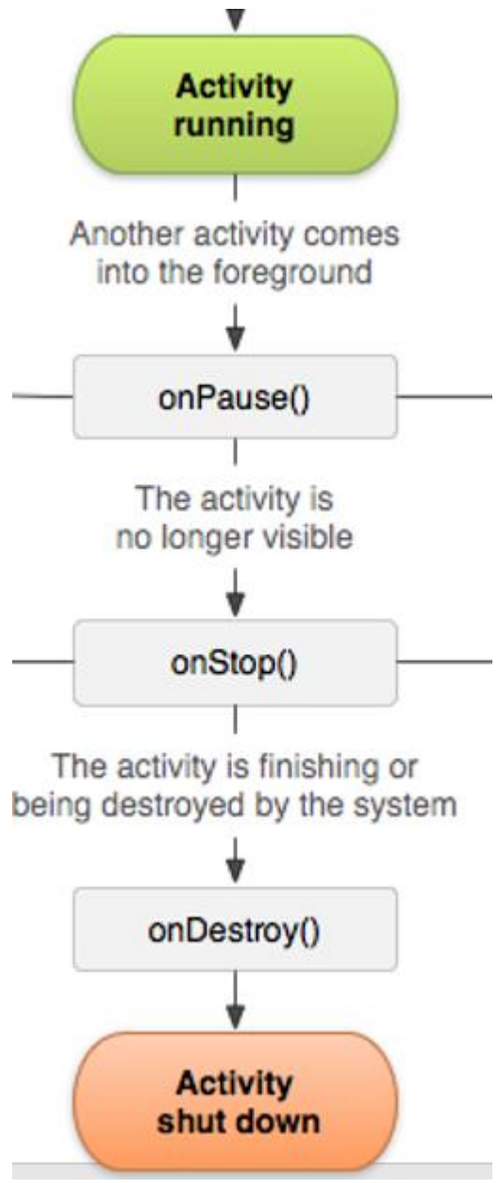
- **onStart()**
  - As onCreate() exits, the activity enters the Started state, and the activity **becomes visible to the user**. This callback contains what amounts to the activity's final preparations for coming to the foreground and becoming interactive



- **onResume()**
  - The system invokes this callback **just before the activity starts interacting** with the user.
  - At this point, the activity is at the top of the activity stack, and captures all user input. Most of an app's core functionality is implemented in the onResume() method

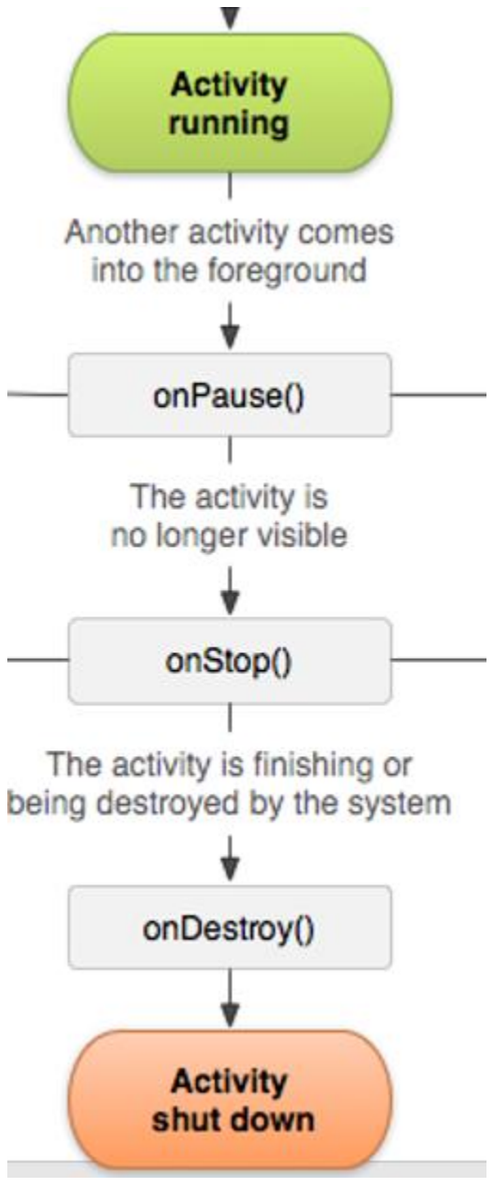
OnPause and OnStop

- **onPause()**
  - The **onPause()** callback always follows **onResume().**
  - The system calls **onPause()** when the activity loses focus and enters a Paused state.
  - This state occurs when, for example, the user taps the Back or Overlay button.
  - When the system calls **onPause()** for your activity, it technically means your activity is still partially visible, but most often is an indication that the user is leaving the activity, and the activity will soon enter the Stopped or Resumed state.



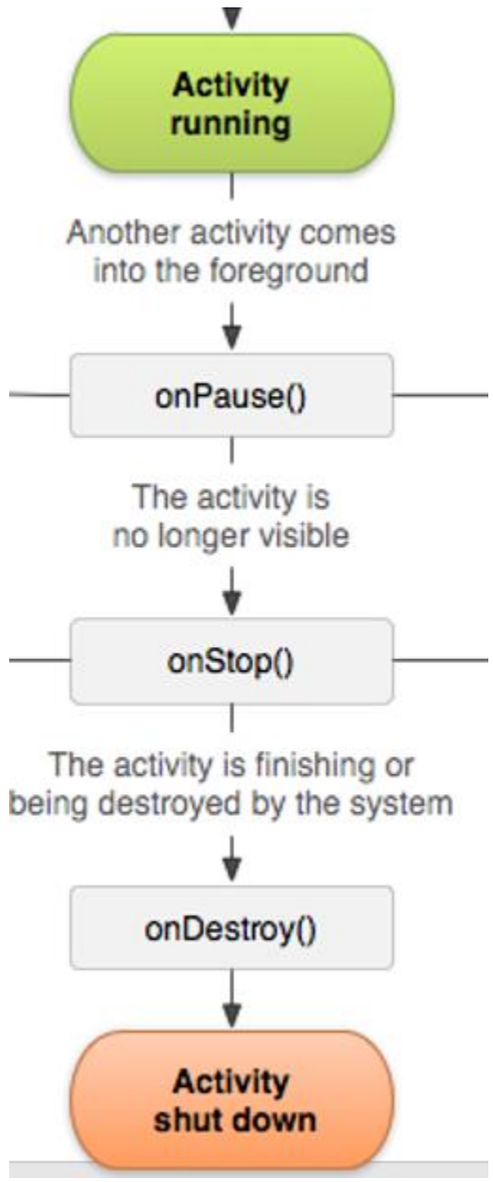
OnPause and OnStop

- **onPause()**
  - An activity in the **Paused state** may continue to update the UI if the user is expecting the UI to update.
  - Examples of **such an activity include one showing a navigation map screen or a media player playing.** Even if such activities lose focus, the user expects their UI to continue updating



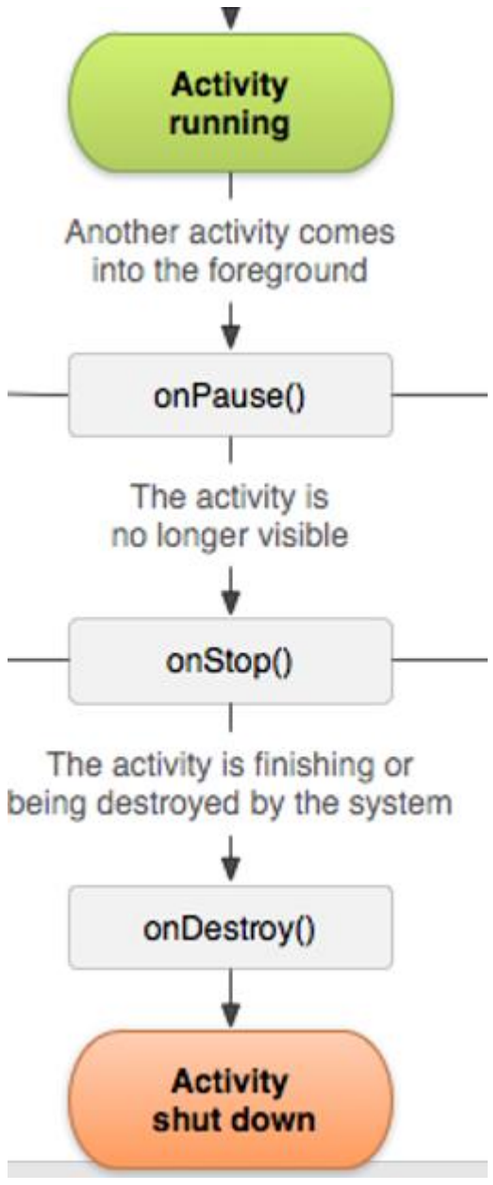
ctd

- You **should not use onPause()** to do **long operations**
  - e.g. save application or user data, make network calls, or execute database transactions
    - as the activity could be killed before they finish
- Once **onPause()** finishes executing, the next callback is either **onStop()** or **onResume()**



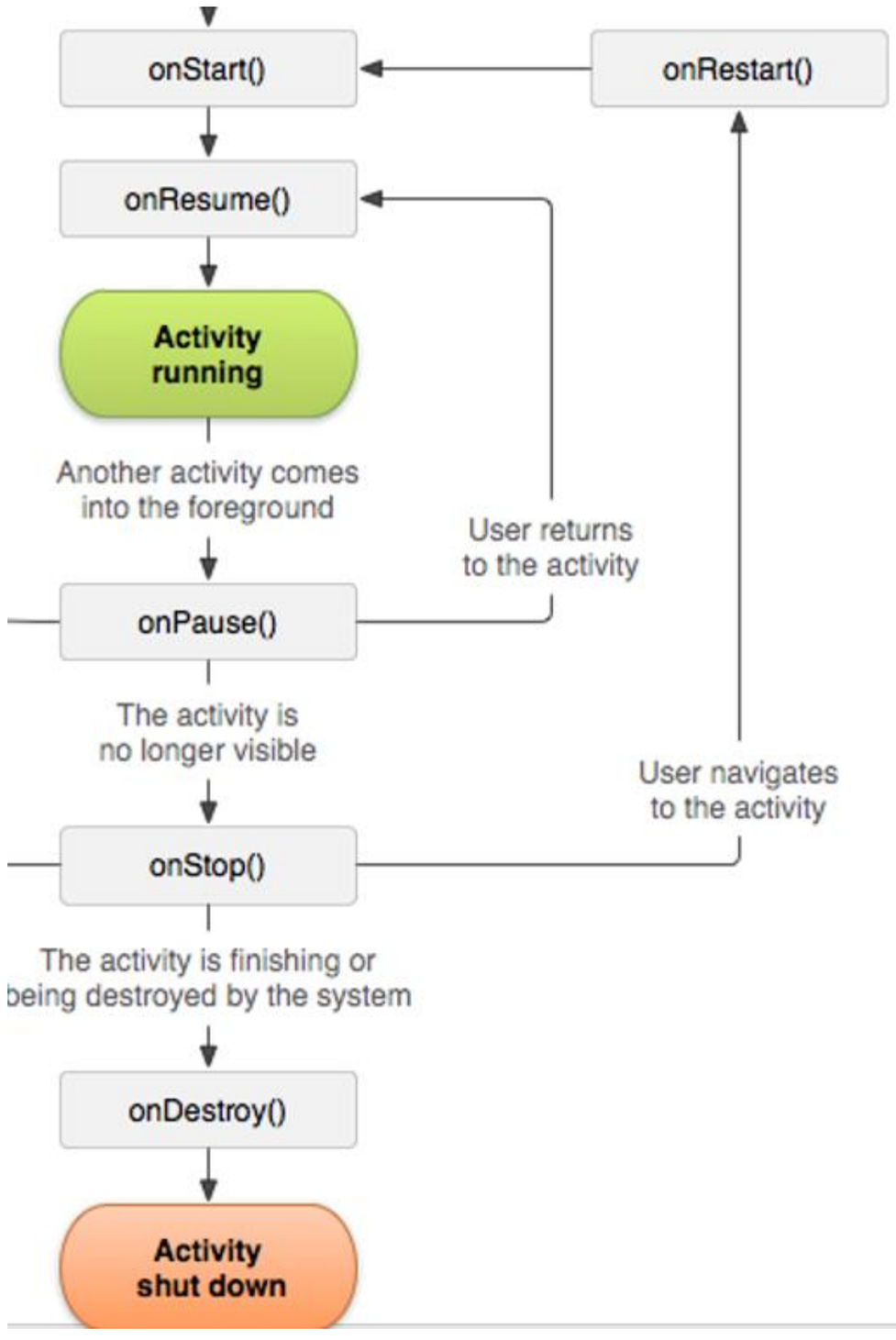
ctd

- **onStop()**
  - The system calls **onStop()** **when the activity is no longer visible** to the user. This may happen because the activity is being destroyed, a new activity is starting, or an existing activity is entering a Resumed state and is covering the stopped activity.
  - In all of these cases, **the stopped activity is no longer visible at all**.
  - The next callback that the system calls is either **onRestart()**, if the activity is coming back to interact with the user, or by **onDestroy()** if this activity is completely terminating.



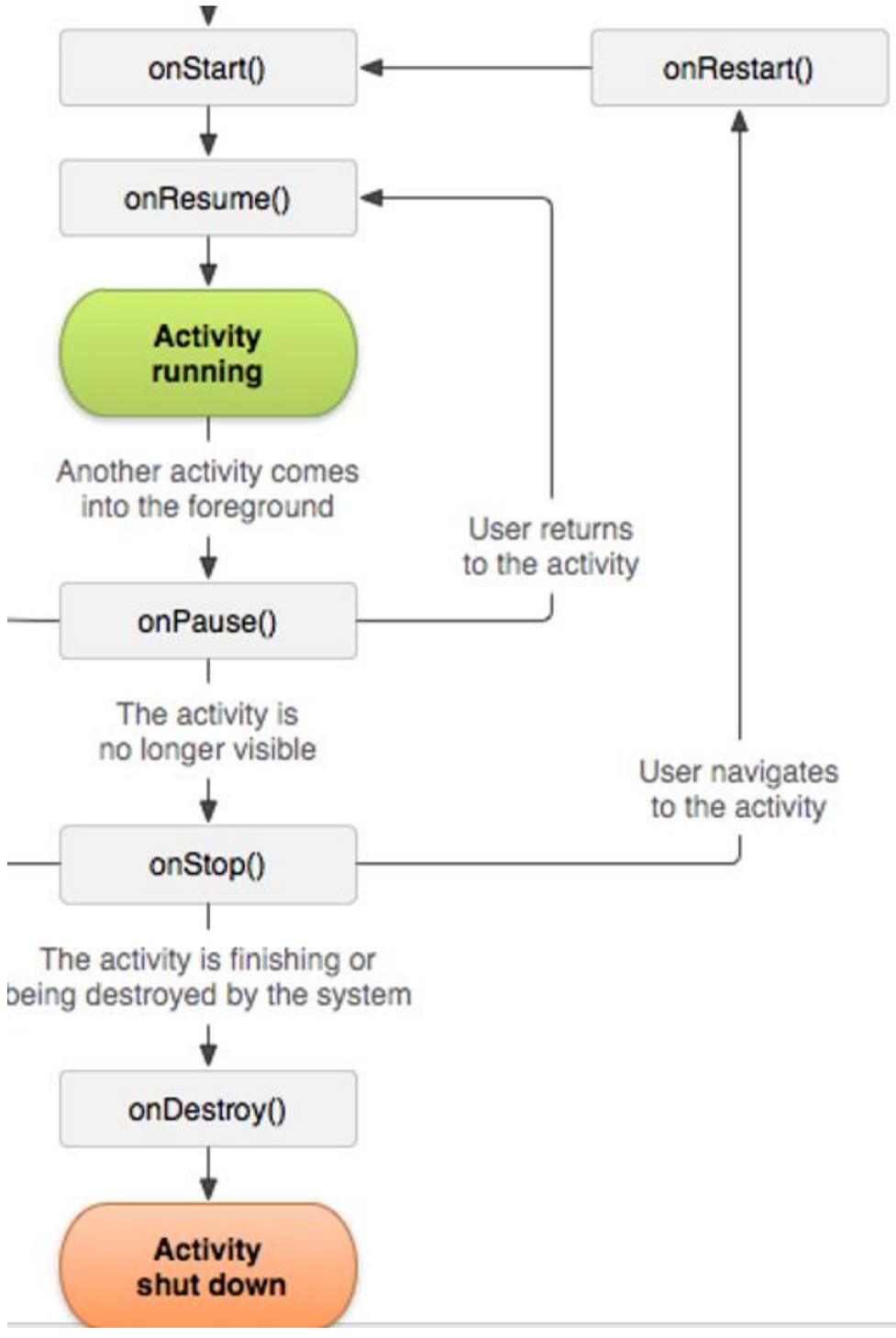
ctd

- **onRestart()**
  - The system invokes this callback when an activity in the Stopped state is about to restart.
  - **onRestart()** restores the state of the activity from the time that it was stopped.
- This callback is always followed by **onStart().**



ctd

- **onDestroy()**
  - The system invokes this callback before an activity is destroyed.
  - This callback is the final one that the activity receives.
  - **onDestroy()** is usually implemented to ensure that all of an activity's resources are released when the activity, or the process containing it, is destroyed.



Entire Lifetime

- The entire lifetime of an activity happens between the first call to onCreate(Bundle) through to a single final call to onDestroy().
  - An activity will do **all setup of "global" state in onCreate**(), and **release all remaining resources in onDestroy()**
    - never leave resources open after onDestroy
      - you will get an error
- For example, if it has a thread running in the background to download data from the network, it may create that thread in onCreate() and then stop the thread in onDestroy().

https://developer.android.com/reference/android/app/Activity.html#ActivityLifecycle

Visible Lifetime

- The **visible lifetime of an activity** happens between a call to **onStart()** until a corresponding call to **onStop().**
  - During this time the user **can see the activity on-screen**, though it may not be in the foreground and interacting with the user.
  - Between these two methods you can maintain resources that are needed to show the activity to the user.
  - For example, you can register a **BroadcastReceiver** in **onStart()** to monitor for changes that impact your UI, and unregister it in **onStop()** when the user no longer sees what you are displaying.
  - The onStart() and onStop() methods **can be called multiple times**, as the activity becomes visible and hidden to the user.

https://developer.android.com/reference/android/app/Activity.html#ActivityLifecycle

Foreground Lifetime

- The **foreground lifetime of an activity** happens between a call to **onResume()** until a corresponding call to **onPause()**.
  - During this time the activity is in front of all other activities and interacting with the user.
- An activity can frequently go between the **resumed** and **paused** states
  - for example when the device goes to sleep, when an activity result is delivered, when a new intent is delivered -- so the **code in these methods should be fairly lightweight**

https://developer.android.com/reference/android/app/Activity.html#ActivityLifecycle

In Kotlin

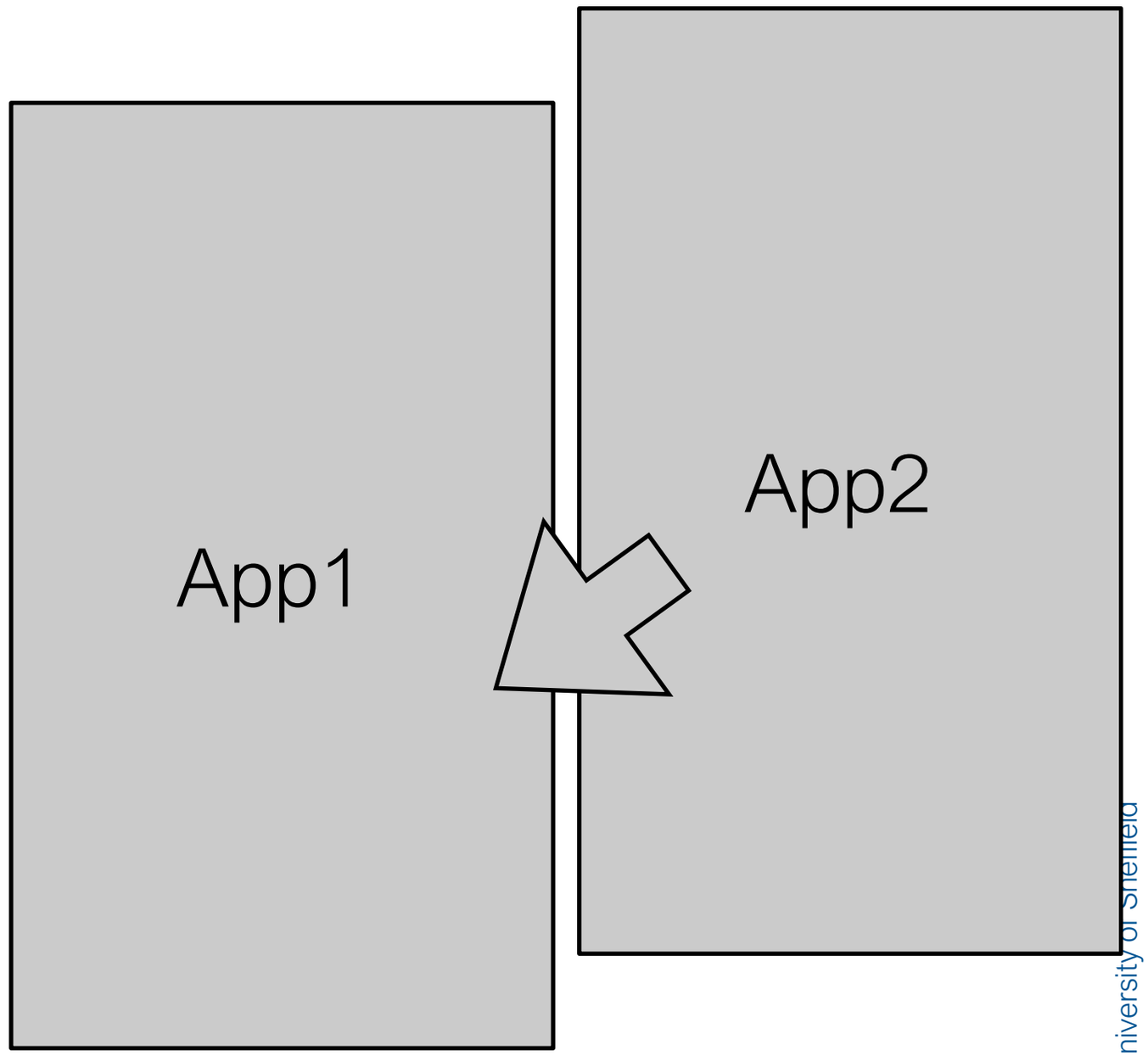
```
class MainActivity : AppCompatActivity() {

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

    override fun onRestart() {...}

    override fun onResume() {...}

    override fun onPause() {...}

    override fun onStop() {...}

    override fun onDestroy() {...}
}
```

As you notice, there is no main method

The University Of Sheffield.

onPause/onResume

Activity is partially obscured



onStop/onStart

App is minimised or superseded by another app

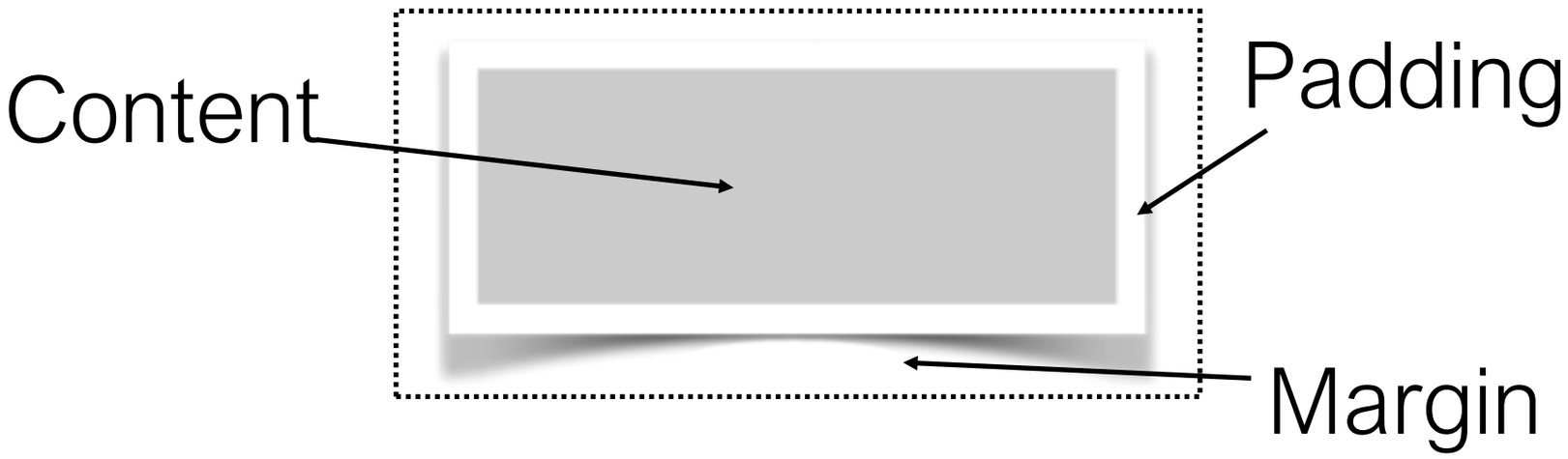
Designing an app with Material Design

Layout as a tree

- The **Android Layout** is a nested tree of elements
  - Called the view hierarchy
  - Root: a view
  - Leaves: the single elements such as buttons, textViews, etc.
- Anything that you see of your app is a single view in this hierarchy

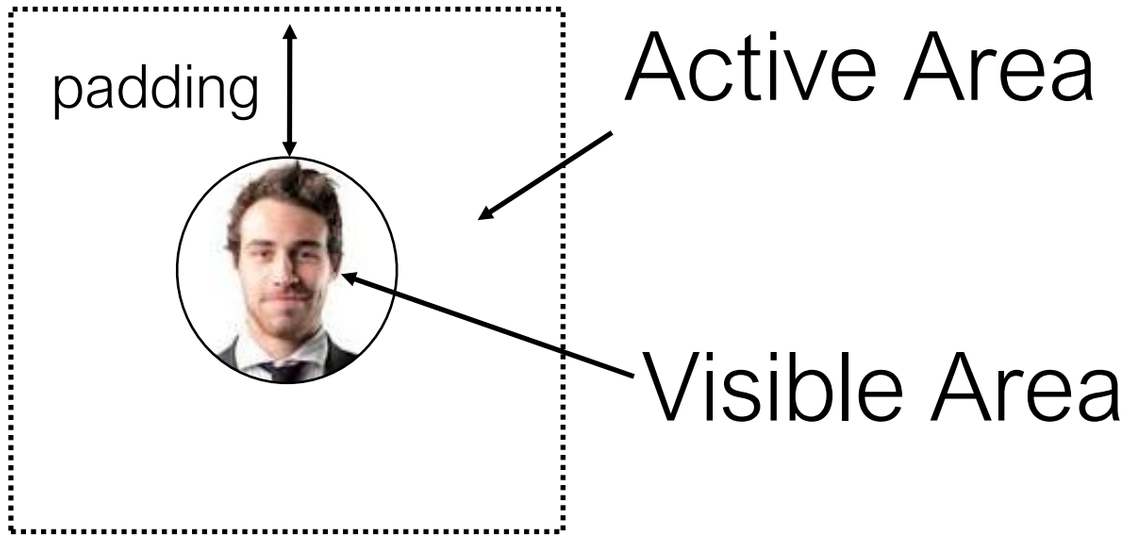
Box view of elements

- Android follows the same box-view used in CSS
  - Every element is contained in a rectangular box
    - even if circular, the element will be represented as the minimum box enclosing the element



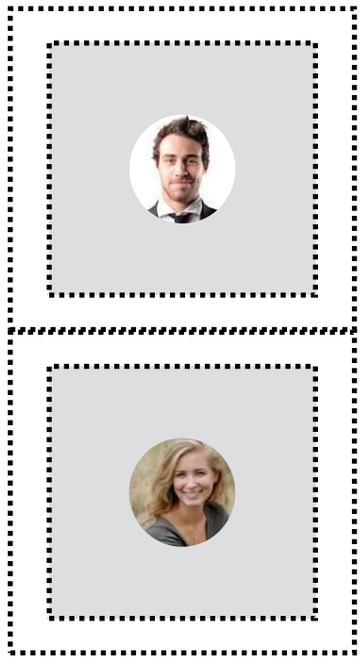
Paddings

- **Padding**:
  - empty space internal to the element
    - typically you will use padding when you want the element to be bigger (e.g. to the touch) than the element actually inside
      - for example to make easier to tap an image that is small, you add a large padding so that it becomes easier to tap



Margins

- **Margin** defines the distance between the current element and the other elements
  - use it to keep your element separated from others
    - i.e. the margin will not be tappable



In Grey the padding (touchable)
in white: the margin (inactive)

Layout Views

Linear Layout

A layout that organizes its children into a single horizontal or vertical row. It creates a scrollbar if the length of the window exceeds the length of the screen.

Relative Layout

Enables you to specify the location of child objects relative to each other (child A to the left of child B) or to the parent (aligned to the top of the parent).

Web View

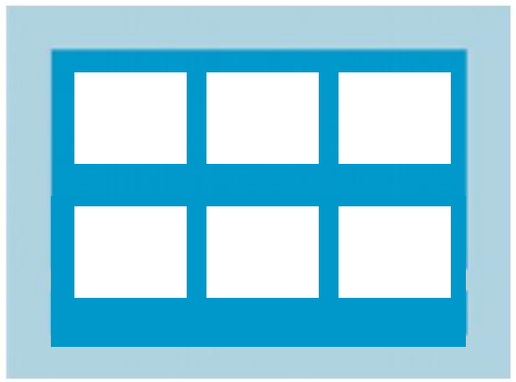
Displays web pages.

Layouts

Frame Layout

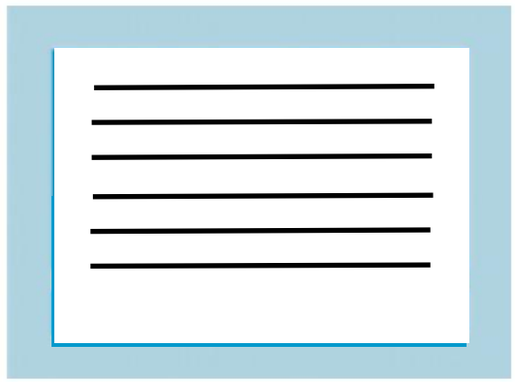
Enables you to specify overlapping element

Grid Layout



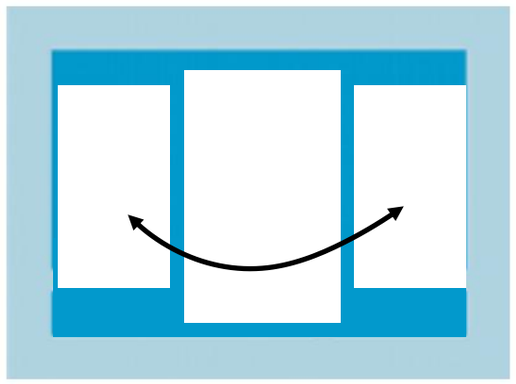
Enables you to specify a grid of elements loaded dynamically (i.e. in Java)
Normally combined with a RecyclerView

ScrollView



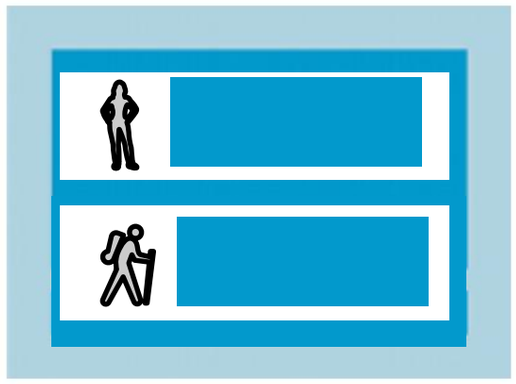
It contains only one element and makes it automatically scrollable

ViewPager



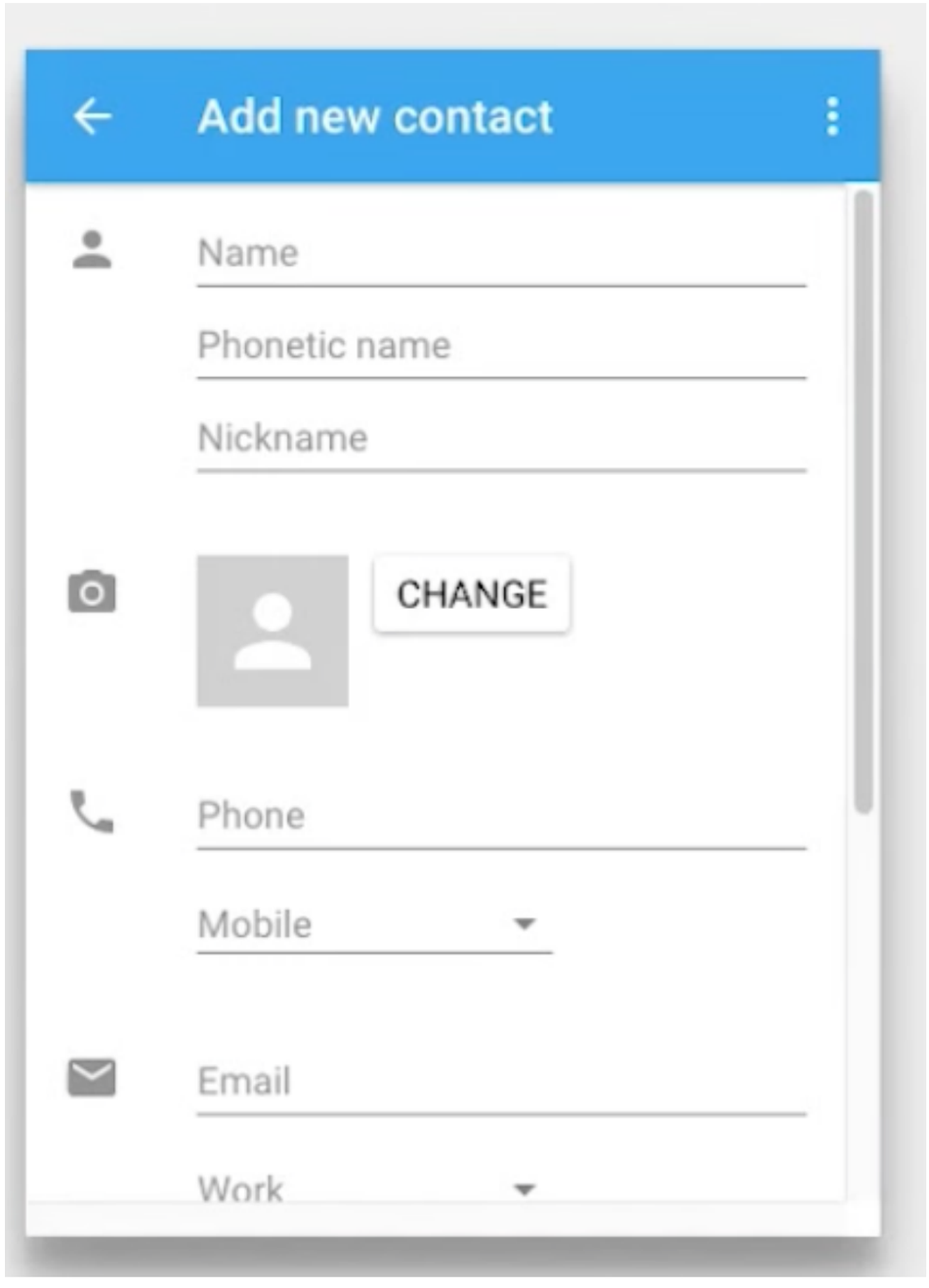
it enables creating horizontally sliding views (e.g. to support tabbed views)

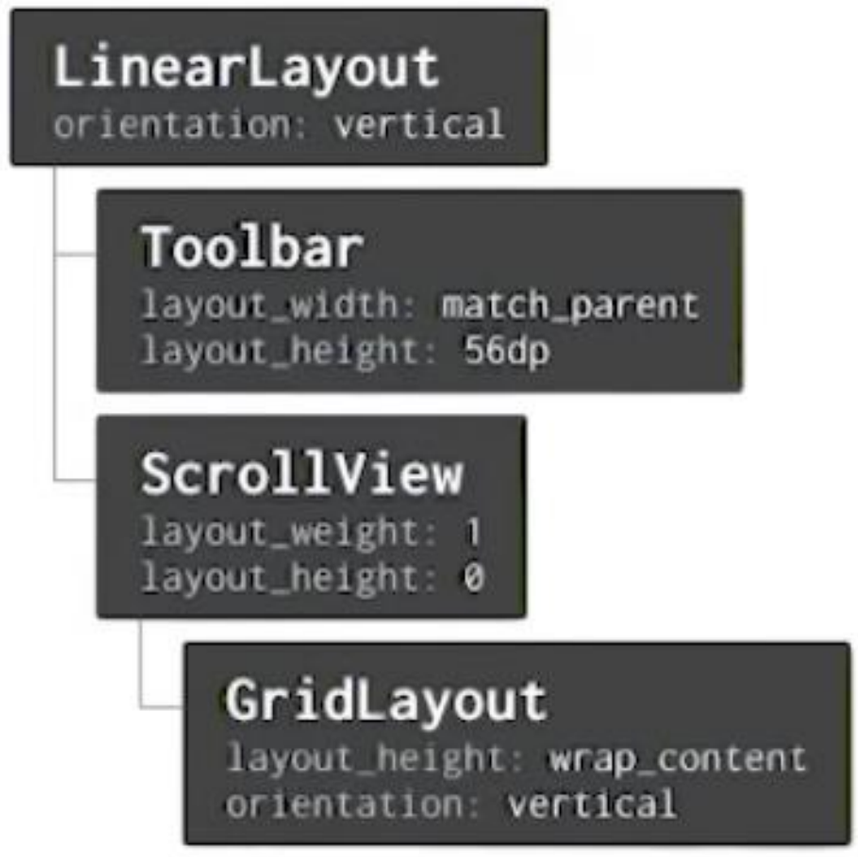
ListView/RecyclerView



It enables creating dynamic lists of elements (complex objects)
Normally combined with a RecyclerView

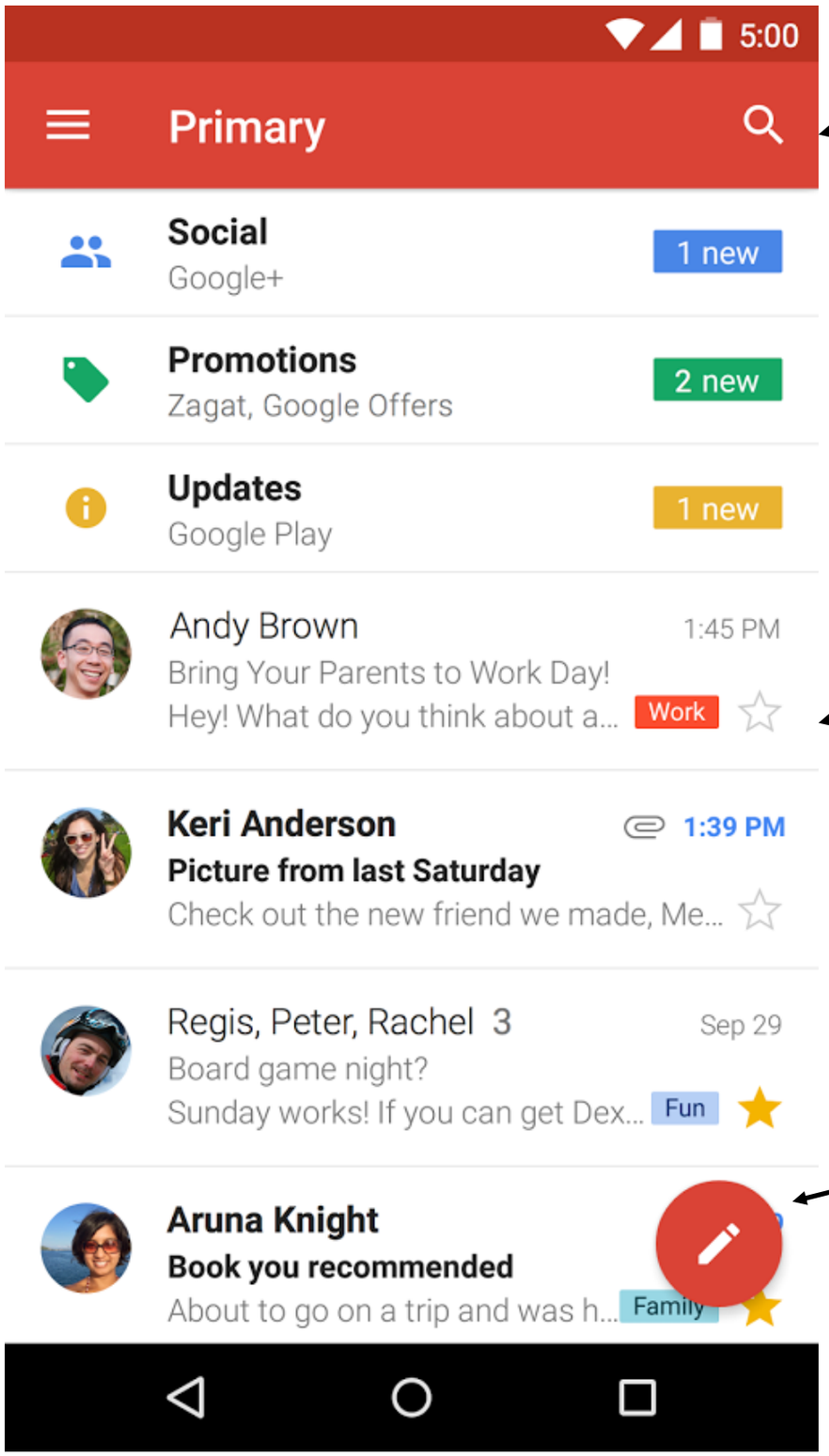
A Simple Example





Grid layout containing
1st columns: icon
2nd column a LinearLayout

Today's Lab Class



Fixed height toolbar with action element

`android:width="match_parent"`
`android:height="56dp"`

Recycler View

(contained in a FrameLayout)
android:width="match_parent"
`android:height="match_parent"`

Horizontal LinearLayout for each row

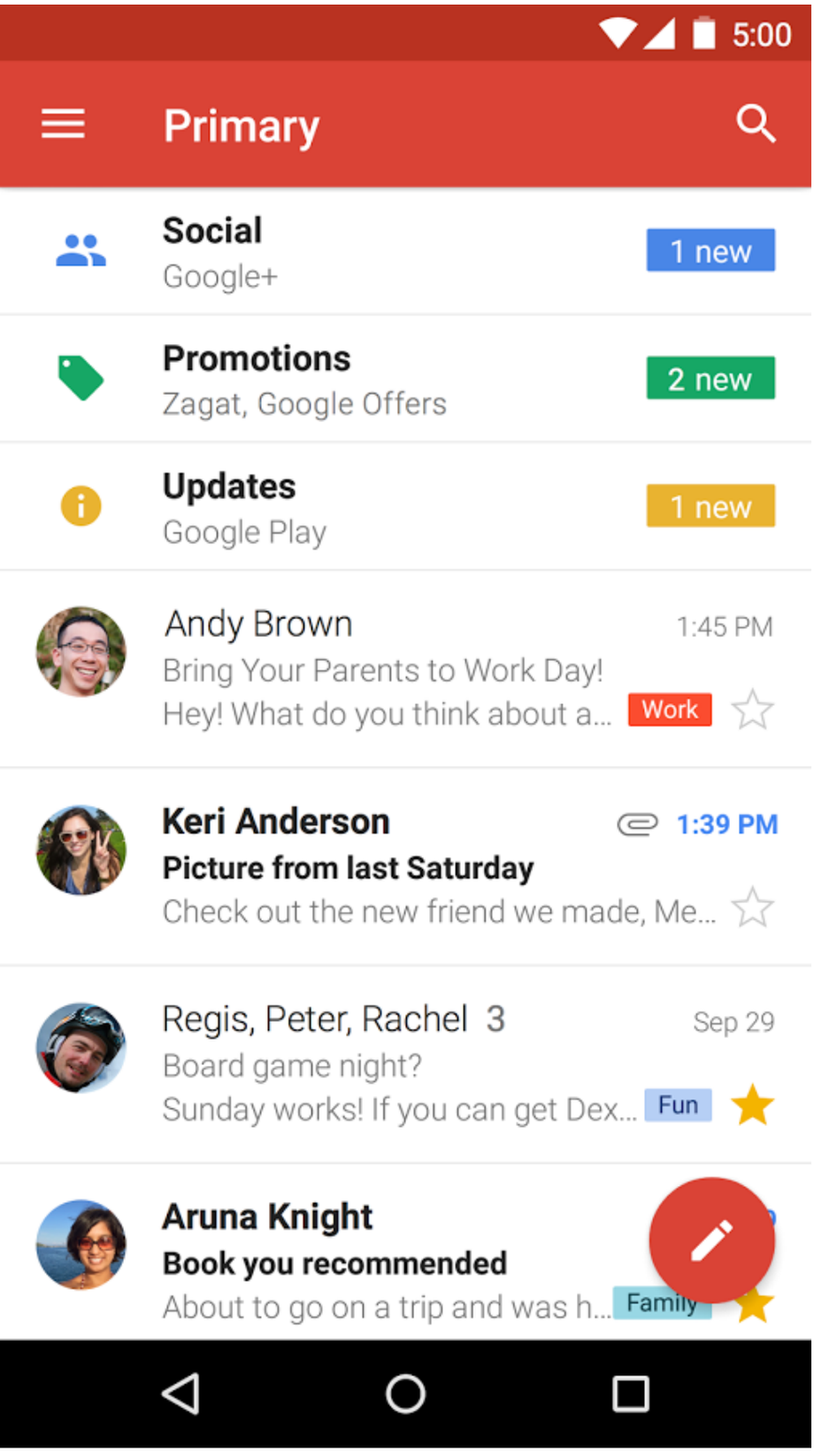
The FrameLayout was inserted to allow overlapping of Floating Button

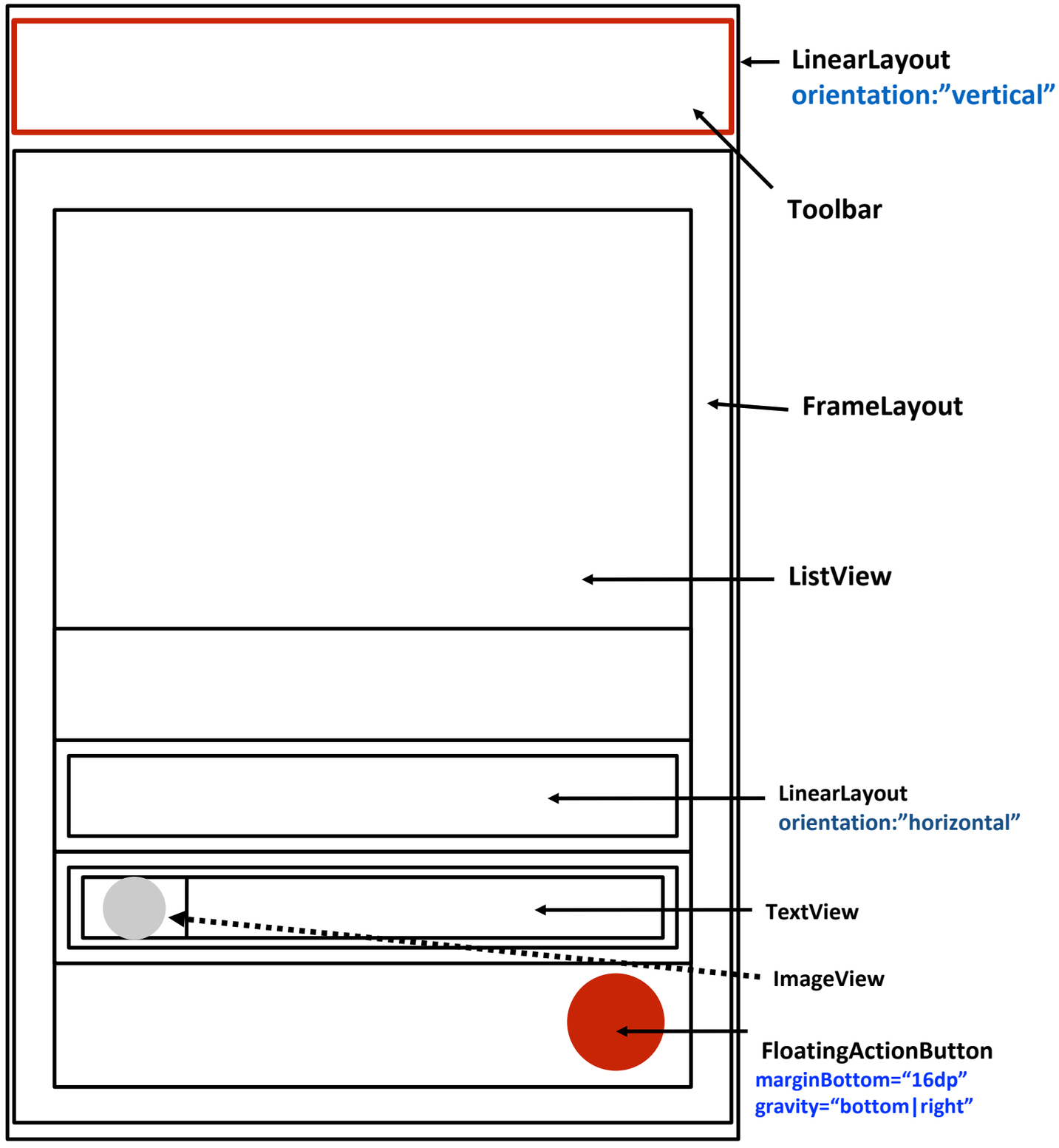
`android:width="match_parent"`
`android:height="0dp"`
`android:weight="1"`

Floating Button

`android:layout_gravity="bottom|right"`

The University Of Sheffield.





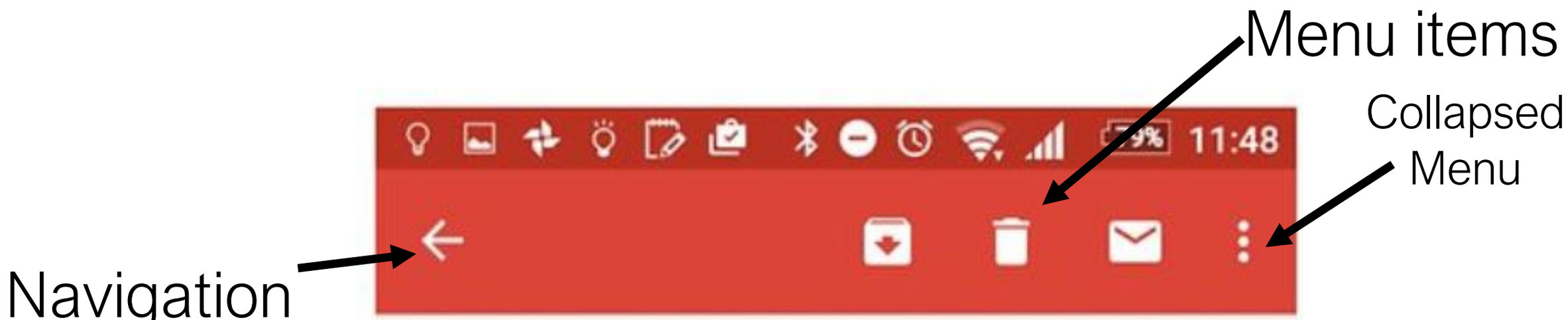
The University Of Sheffield.

Android Navigation

Menus, ActionBar/ToolBar, and Navigation Drawer

Toolbar

- Standard way to present navigation and menu items
  - if sitting at the top of an activity it is called App Bar

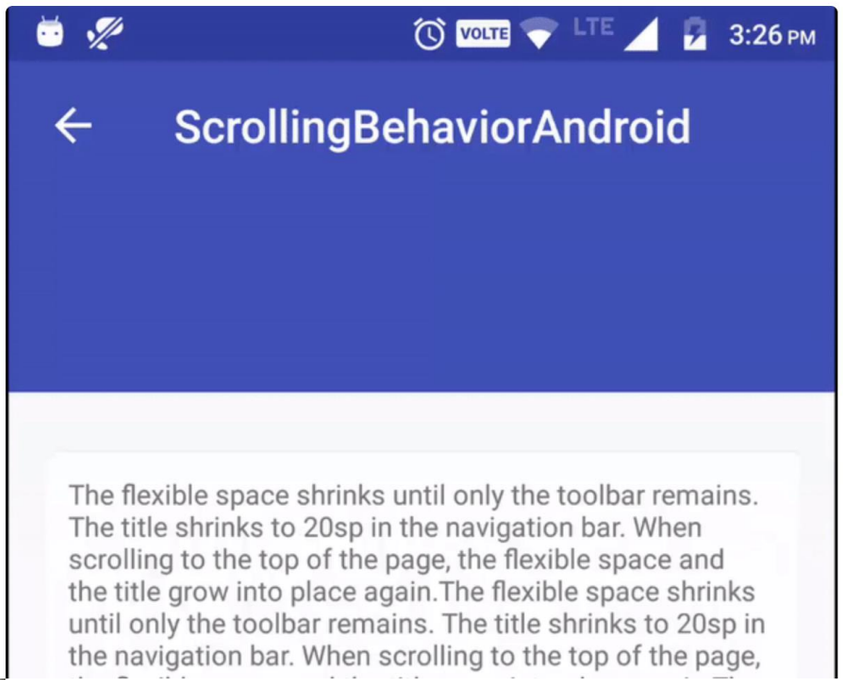


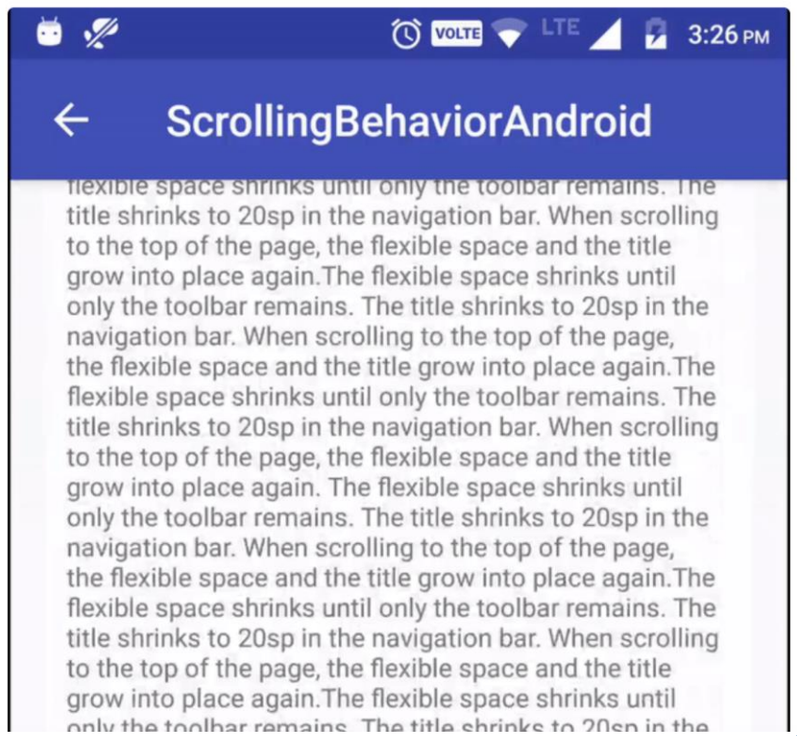

```
<androidx.appcompat.widget.Toolbar
    android:id="@+id/toolbar"
    android:layout_width="match_parent"
    android:layout_height="?attr/actionBarSize"
    android:background="?attr/colorPrimary"
    app:popupTheme="@style/Theme.MyApplication.PopupOverlay" />
```

```
<android.support.v7.widget.Toolbar

    android:layout_width="match_parent"

    android:layout_height="?attr/actionBarSize
```

Extended Appbar

- Used to show logos and branding
- May be collapsable if above a scrollview





http://karthikraj.net/2016/12/24/scrolling-behavior-for-appbars-in-android/

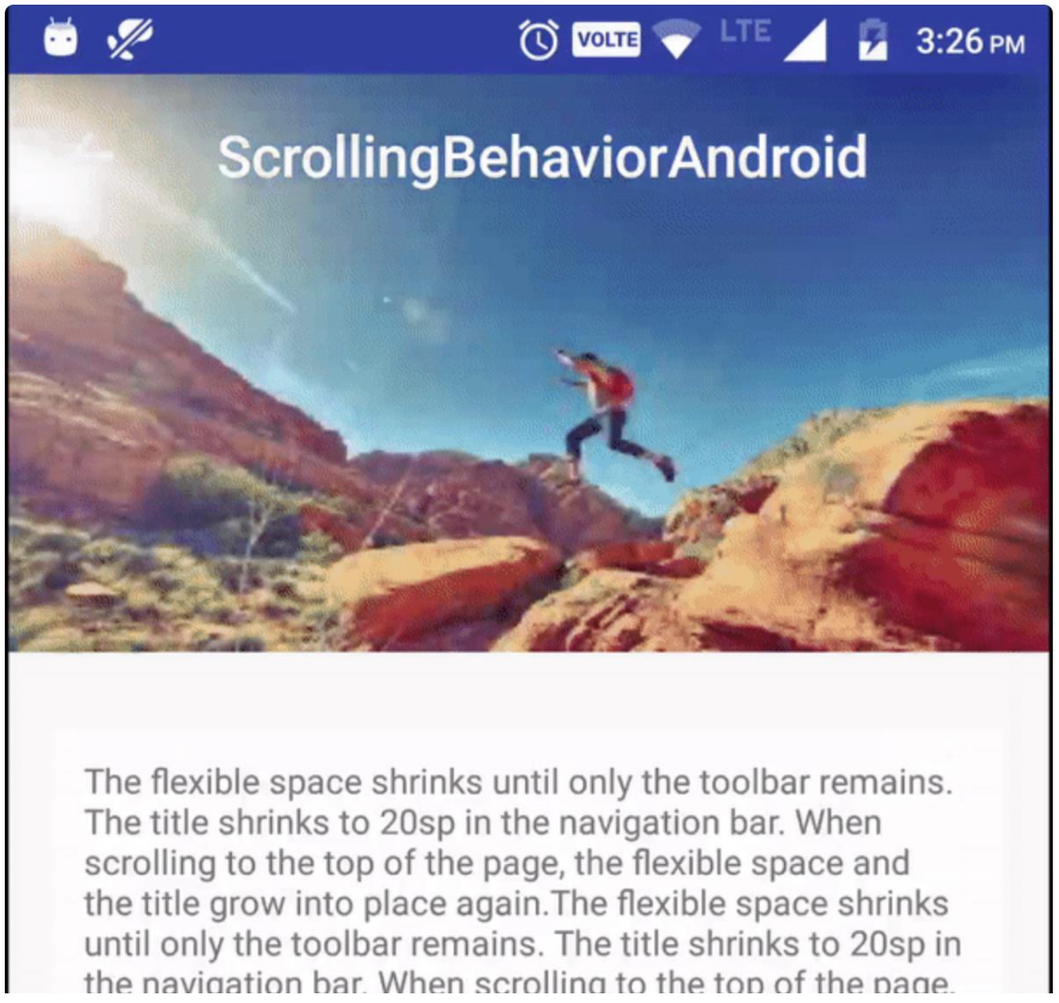
Scrolling up

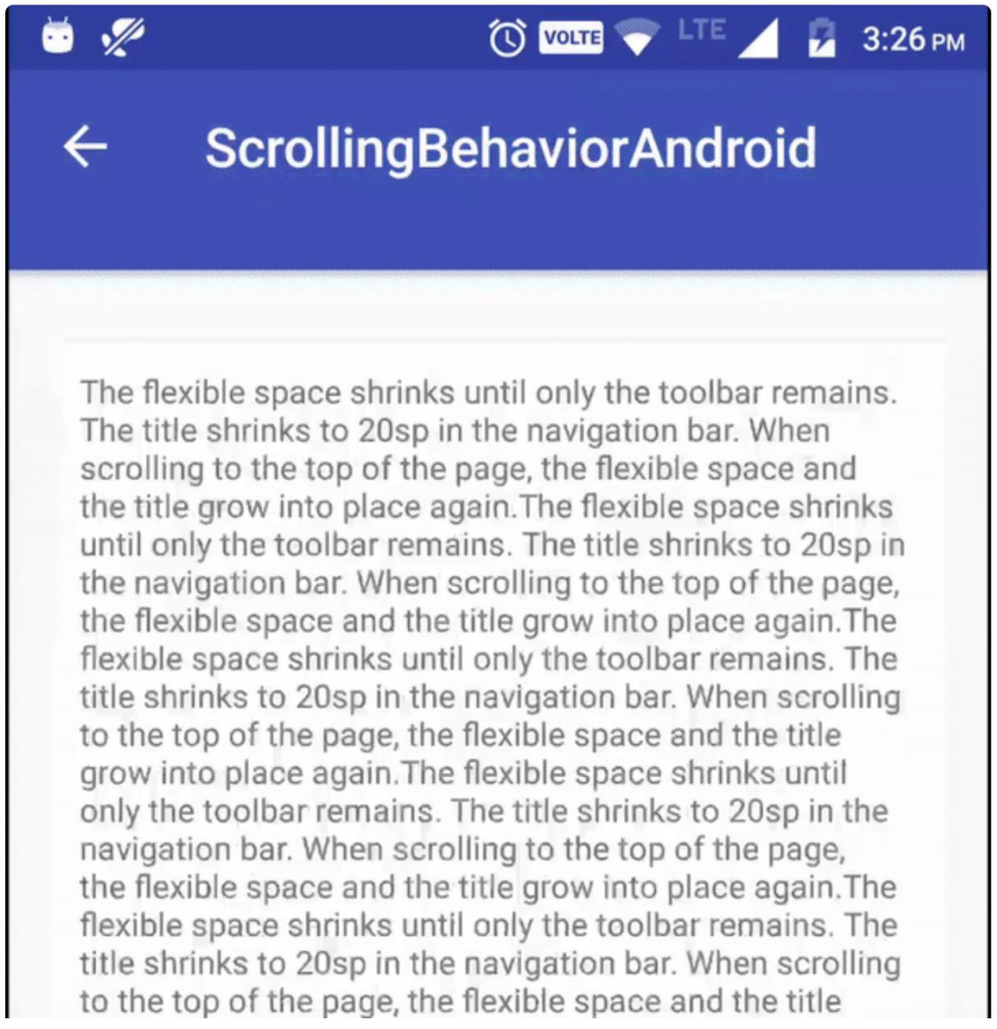
In XML

```
<com.google.android.material.appbar.AppBarLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:theme="@style/Theme.MyApplication.AppBarOverlay">

    <com.google.android.material.appbar.CollapsingToolbarLayout
        android:id = "@+id/collapsing_toolbar_layout"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:contentScrim="?attr/colorPrimary"
        app:layout_scrollFlags="scroll|exitUntilCollapsed"/>

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:background="?attr/colorPrimary"
        app:popupTheme="@style/Theme.MyApplication.PopupOverlay" />

</com.google.android.material.appbar.AppBarLayout>
```

https://medium.com/@ankitashettya/collapsing-toolbar-in-android-using-androidx-jetpack-d54678ac4cbc

Collapsing with Image





Collapsing and Traditional TB

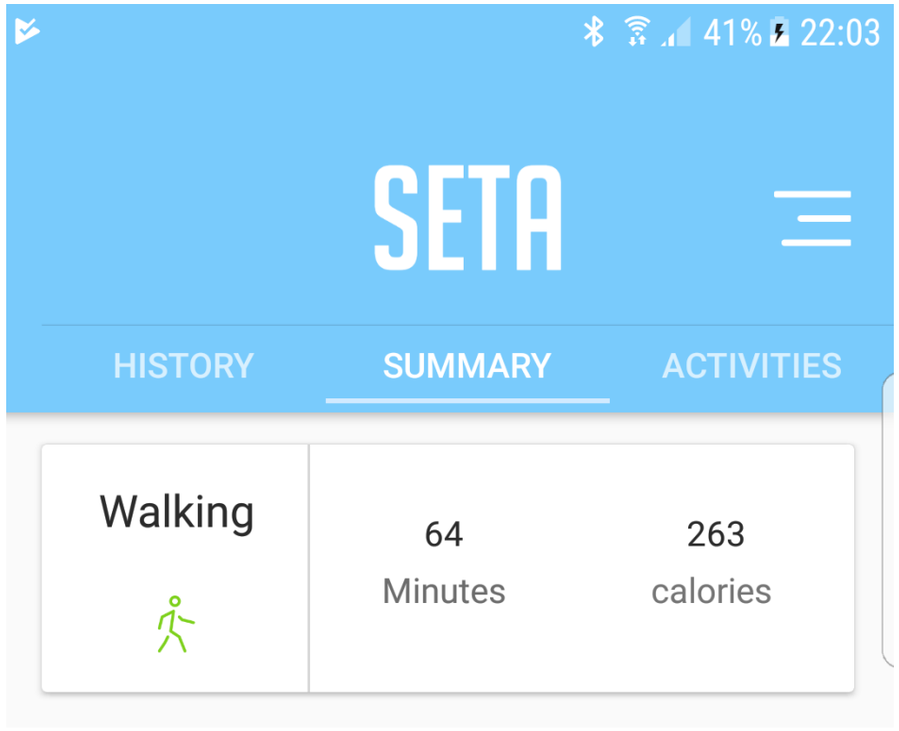
```
<com.google.android.material.appbar.AppBarLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:theme="@style/Theme.MyApplication.AppBarOverlay">

    <com.google.android.material.appbar.CollapsingToolbarLayout
        android:id = "@+id/collapsing_toolbar_layout"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:contentScrim="?attr/colorPrimary"
        app:layout_scrollFlags="scroll|exitUntilCollapsed"/>

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="200dp"
        app:layout_collapseMode="parallax"/>

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:background="?attr/colorPrimary"
        app:popupTheme="@style/Theme.MyApplication.PopupOverlay" />

</com.google.android.material.appbar.AppBarLayout>
```

Collapsing one: will disappear

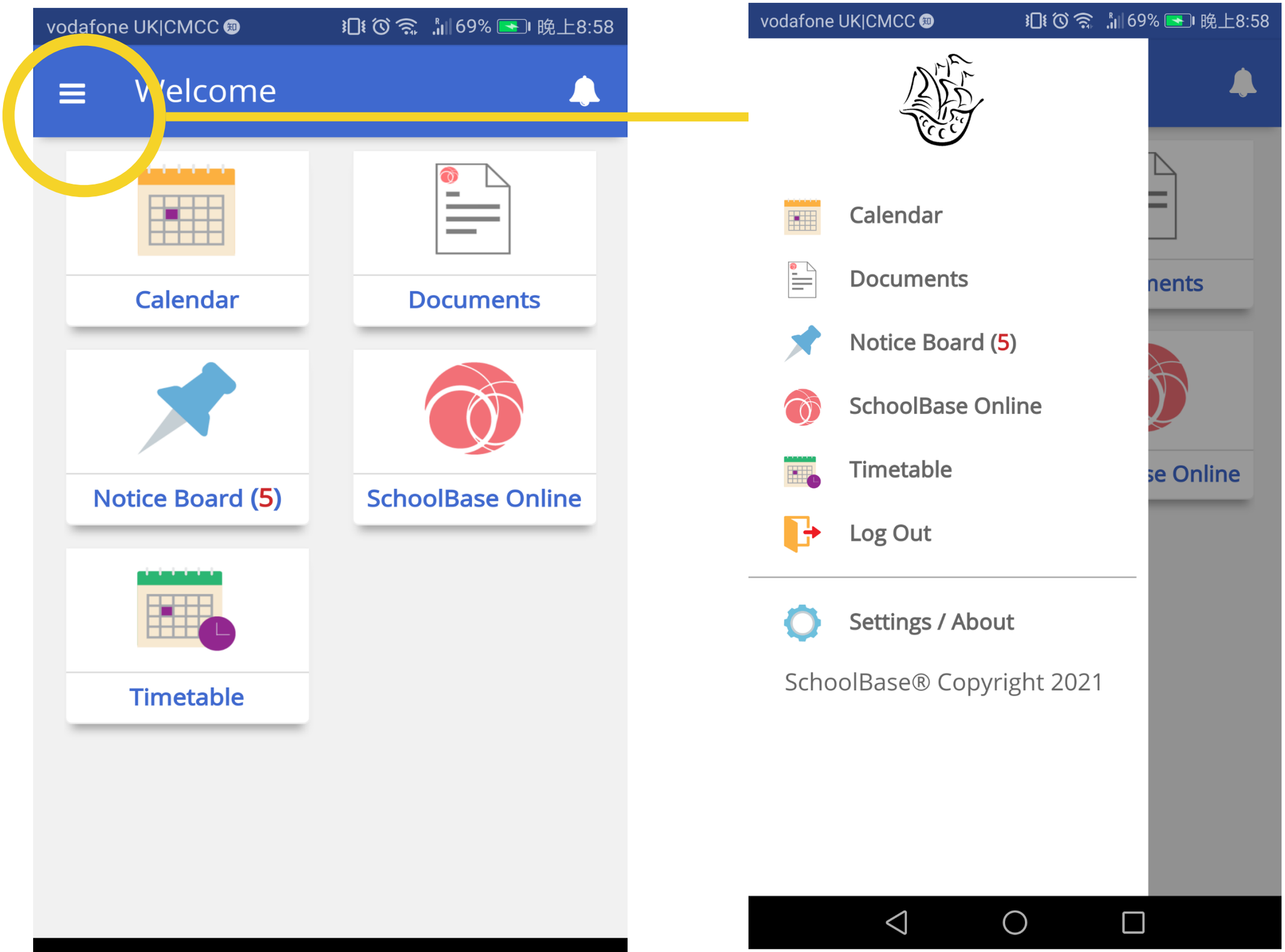
Traditional one: will stay

Tabs

- Allow pagination
- typically coordinated by a ViewPager to slide across pages

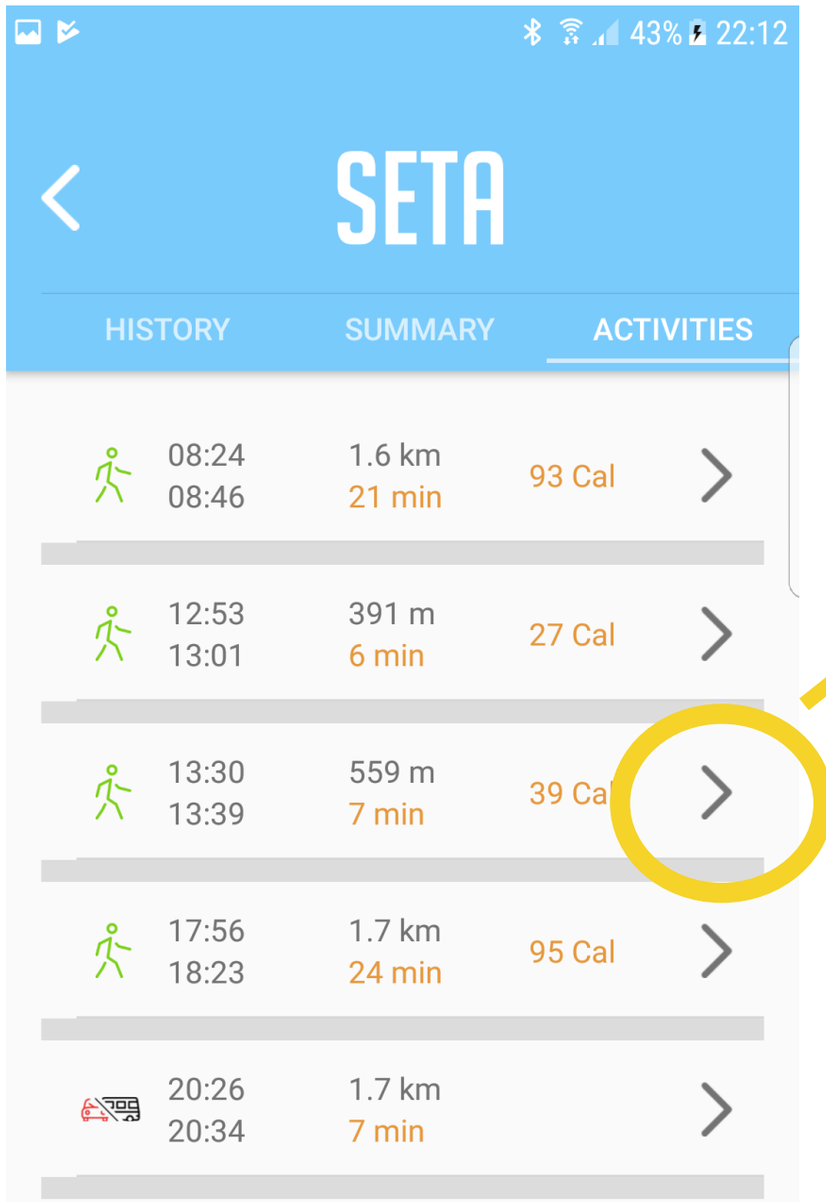


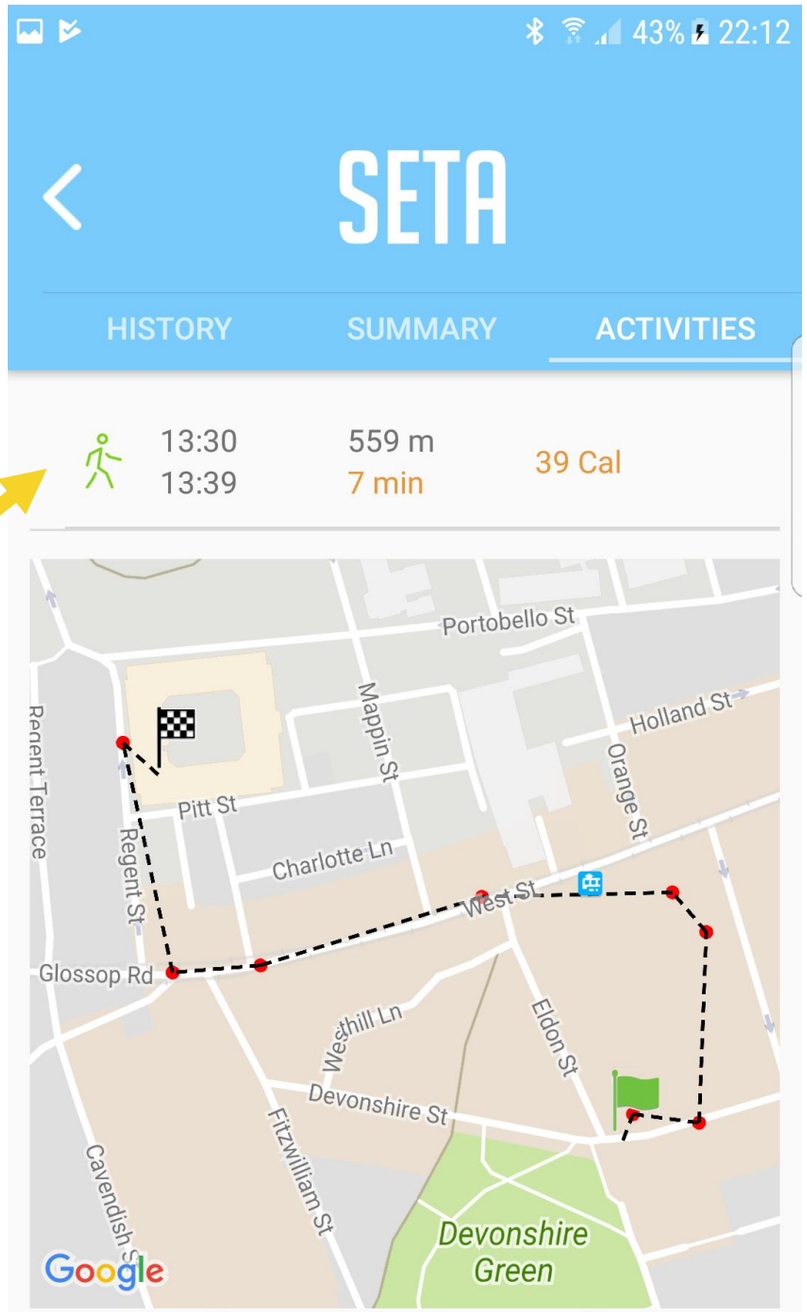
Navigation drawer



Horizontal Scrolling

- Typically to show details of an item in e.g. a list



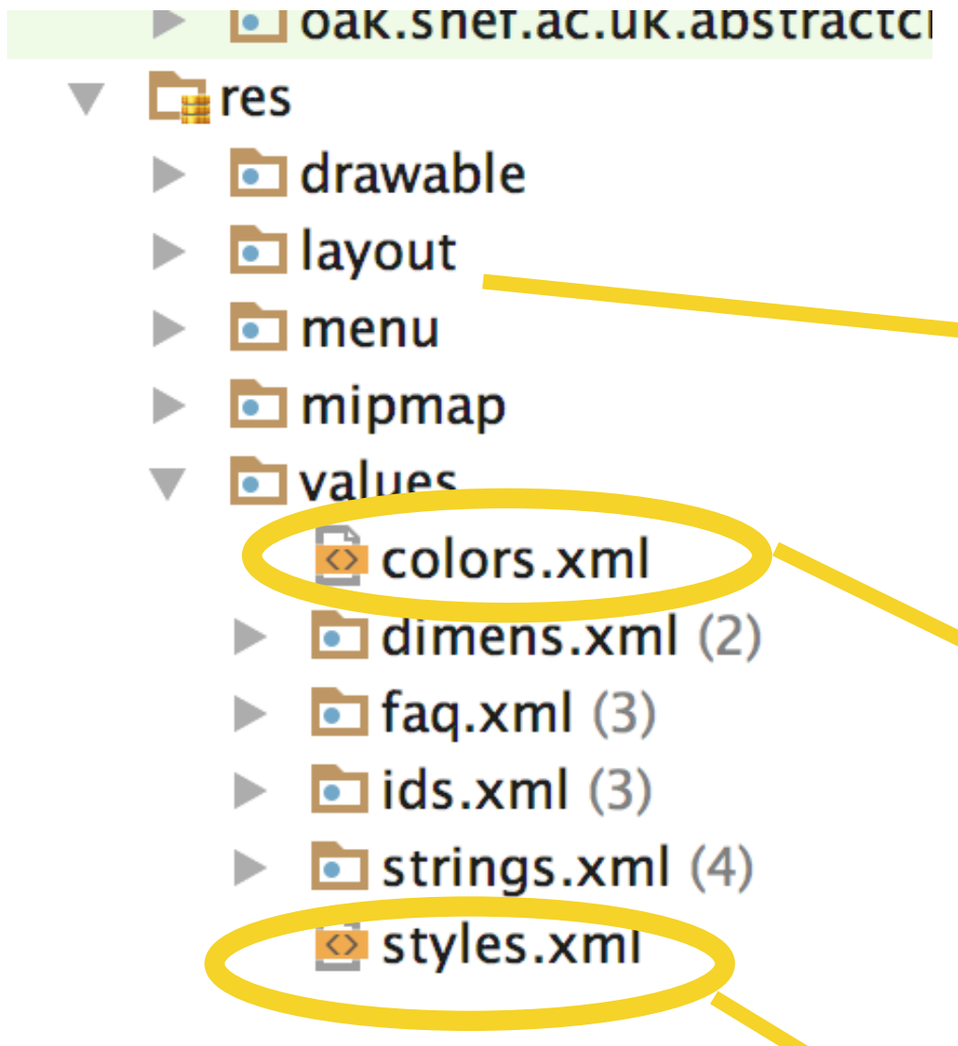


The University Of Sheffield.

Styles and Themes

Coherence

- Applications need to be **coherent** and give a **sense of unity**
  - Branding but also legibility and user comfort
- **Themes**
  - Allow creating a standardisation of look and feel of the entire app
    - it operates on single element (e.g. buttons)
- **Styles**
  - Allow standardising single elements (e.g. buttons)
    - if applied to an element, the style is inherited by its children
    - If applied to the root of the layout it applies to the entire activity
    - If applied to all roots of all activities, it applies to the entire app




```xml
<Button
    android:id="@+id/continue_button"
    android:text="Continue"
    android:enabled="false"
    android:layout_marginTop="8dp"
    android:textAllCaps="false"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    style="@style/Seta_BlueButton"/>
```

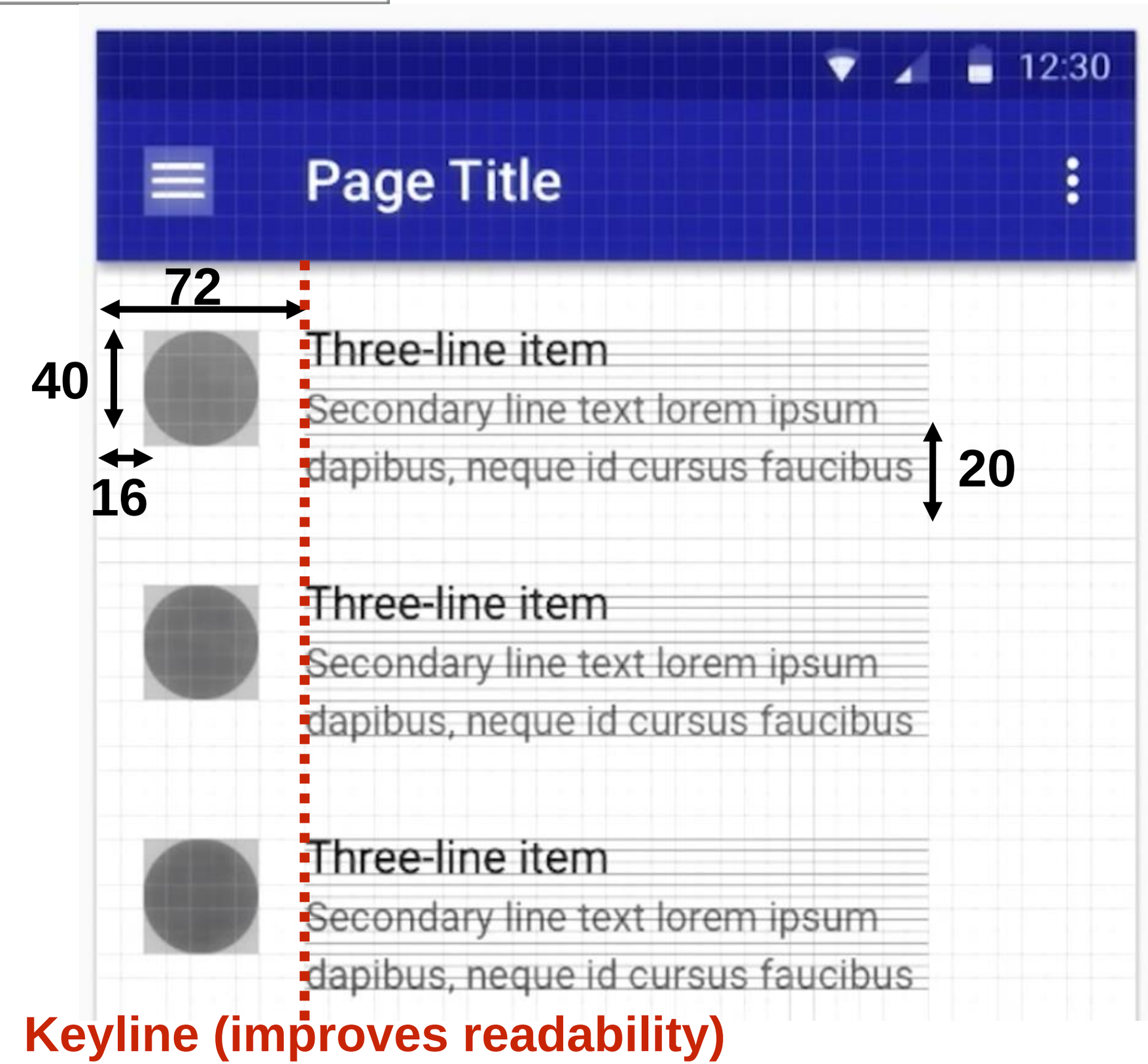
```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#011993</color>
    <color name="colorAccent">#FF4081</color>
</resources>
```

```xml
<!-- Base application theme. -->
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    <!-- Customize your theme here. -->
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
</style>

<style name="Seta_BlueButton">
    <item name="android:background">#6CBDFC</item>
    <item name="android:textColor">#FFF</item>
    <item name="android:textStyle">bold</item>
</style>
```

The University Of Sheffield.

Layout Grids

8dp grid



- Materials uses 8dp grid for components (4dp for text)
- All sizes and distances are multiples of 8 and 4

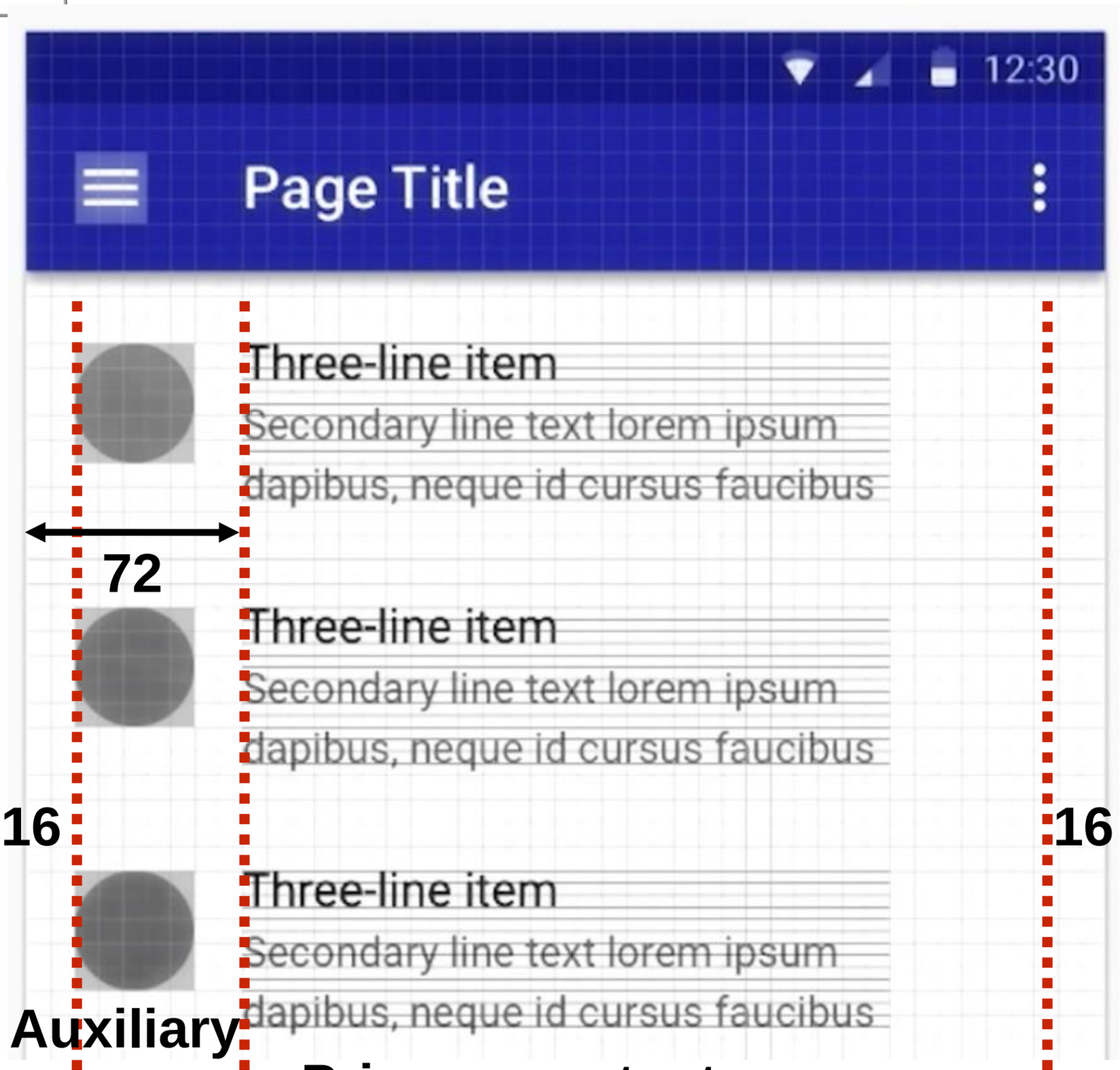
dp?

- dp= the conversion of dp units to screen pixels is simple: px = dp * (dpi / 160)
  - For example, on a 240 dpi screen, 1 dp equals 1.5 physical pixels.
  - You should always use dp units when defining your application's UI, to ensure proper display of your UI on screens with different densities
- dpi= the quantity of pixels within a physical area of the screen; usually referred to as dpi (dots per inch).

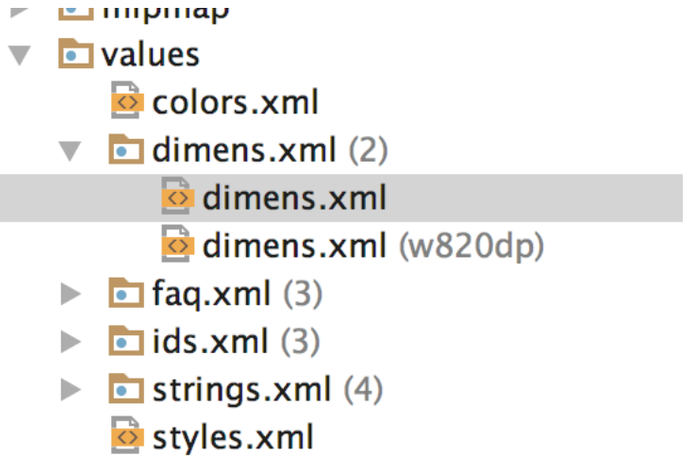
Never use px!!!

https://developer.android.com/guide/practices/screens_support.html

The University Of Sheffield.



The dimen.xml file




```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Default screen margins, per the Android Design guidelines.
    <dimen name="activity_horizontal_margin">16dp</dimen>
    <dimen name="activity_vertical_margin">16dp</dimen>
```

The layout declaration (top view)

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="@dimen/activity_horizontal_margin"
    android:layout_marginRight="@dimen/activity_horizontal_margin"
    tools:ignore="MissingPrefix">
```

Never insert sizes directly into a view, (e.g. android:layout="10dp") - always use a dimension

The University Of Sheffield.

Colours

Limited colours

- Apps must be **consistent** and use fixed limited range of colours
- **Primary colour**
  - the color of large blocks
  - branding
- **Accent colour**
  - the colour of things that must stand up
    - e.g. exceptional actions
- Primary color will also use a palette of variations
  - use a couple of variations of the primary color
    - typically one darker and one lighter

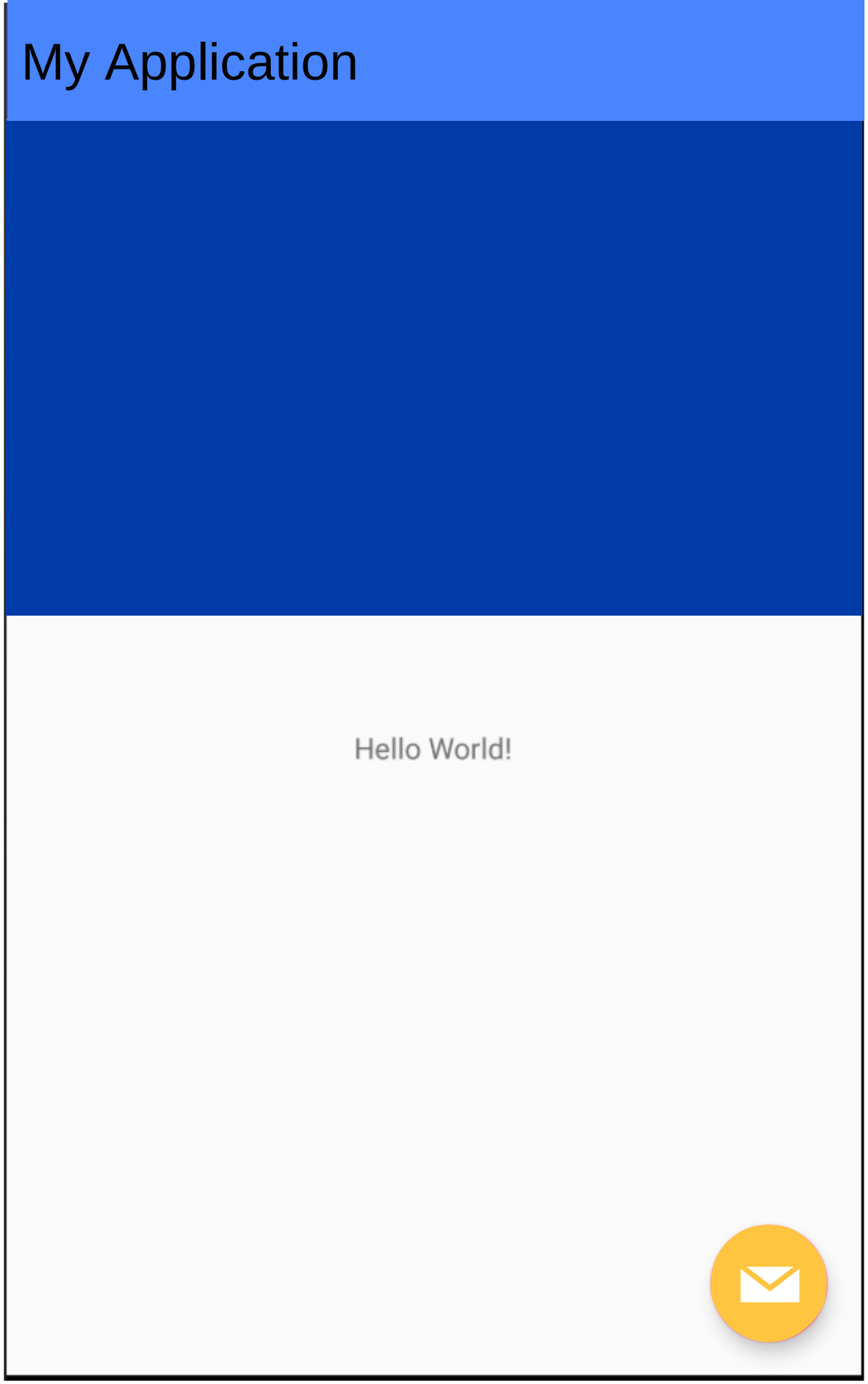
The University Of Sheffield.

Primary color

Primary color palette

Red	
500	#F44336
50	#FFEBEE
100	#FFCDD2
200	#EF9A9A
300	#E57373
400	#EF5350
500	#F44336
600	#E53935
700	#D32F2F
800	#C62828
900	#B71C1C
A100	#FF8A80
A200	#FF5252
A400	#FF1744
A700	#D50000

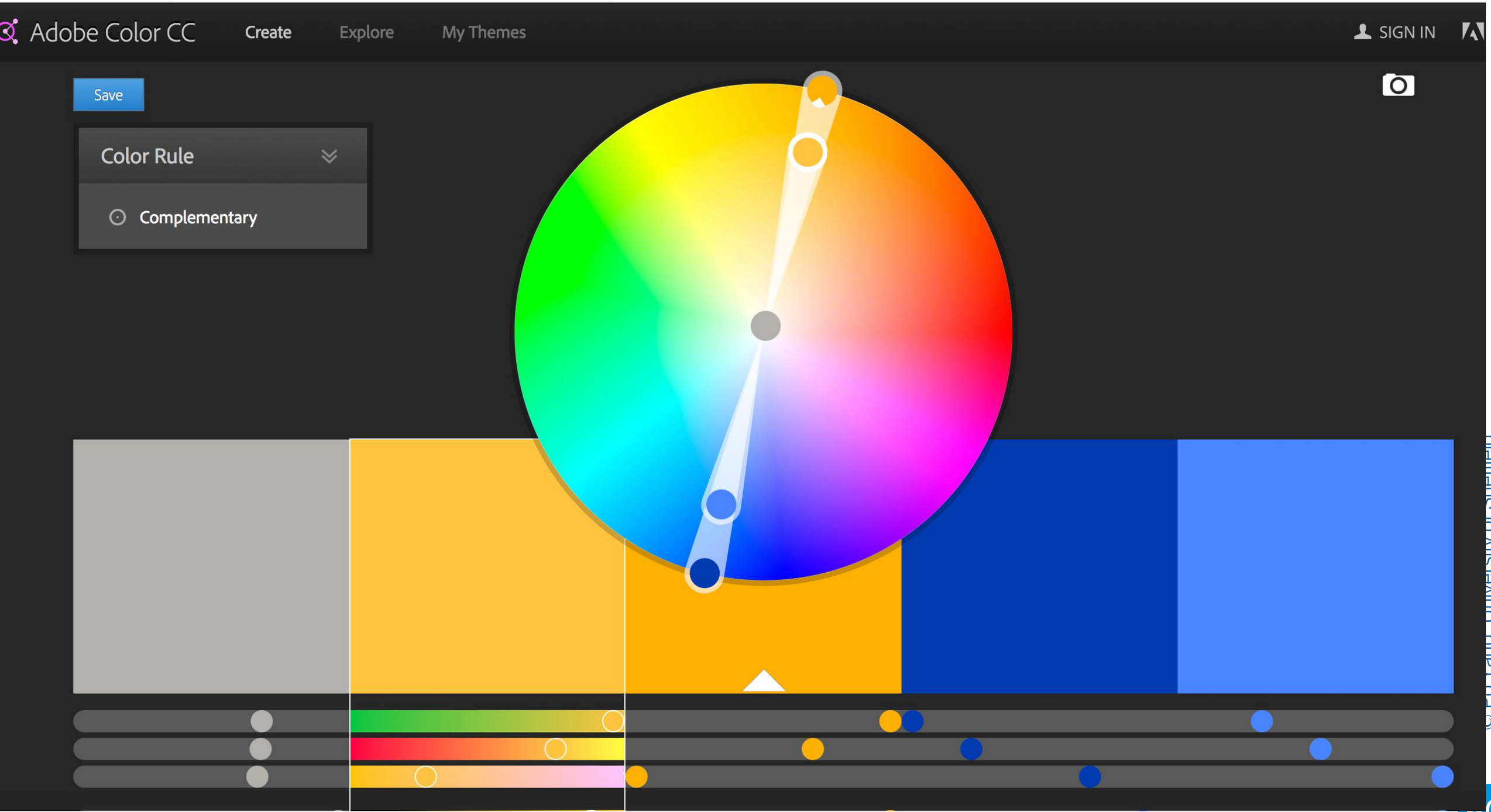
The University Of Sheffield.



The University Of Sheffield.

color.adobe.com

- To choose similar and complementary colours



Summary

- Part 1:
  - Activity's Lifecycle
  - Designing an app with Material Design
- Part 2:
  - Android Navigation
  - Styles and Themes
  - Layout Grids
- Lab tutorial :
  - Designing sensible layouts