COM4510/6510
Software Development for Mobile Devices

Lecture 1: Intro to Mobile Computing & Android

Dr Po Yang

Organisations, Information and Knowledge Group
Department of Computer Science
The University of Sheffield
po.yang@sheffield.ac.uk

Lecture Overview

- Part 1: What is Mobile Computing?
  - A brief history
  - Current trends
- Part 2:What is Android?
  - Architecture
  - App components
  - Entry points and interacting

Mobile computing

- "Ability to use technology 'untethered', that is not physically connected, or in remote or mobile (non static) environments" [Wikipedia]
- Mobile computing involves mobile communication, mobile hardware, and mobile software
- **We will focus on the software aspect for Mobile Devices (Smartphones)**

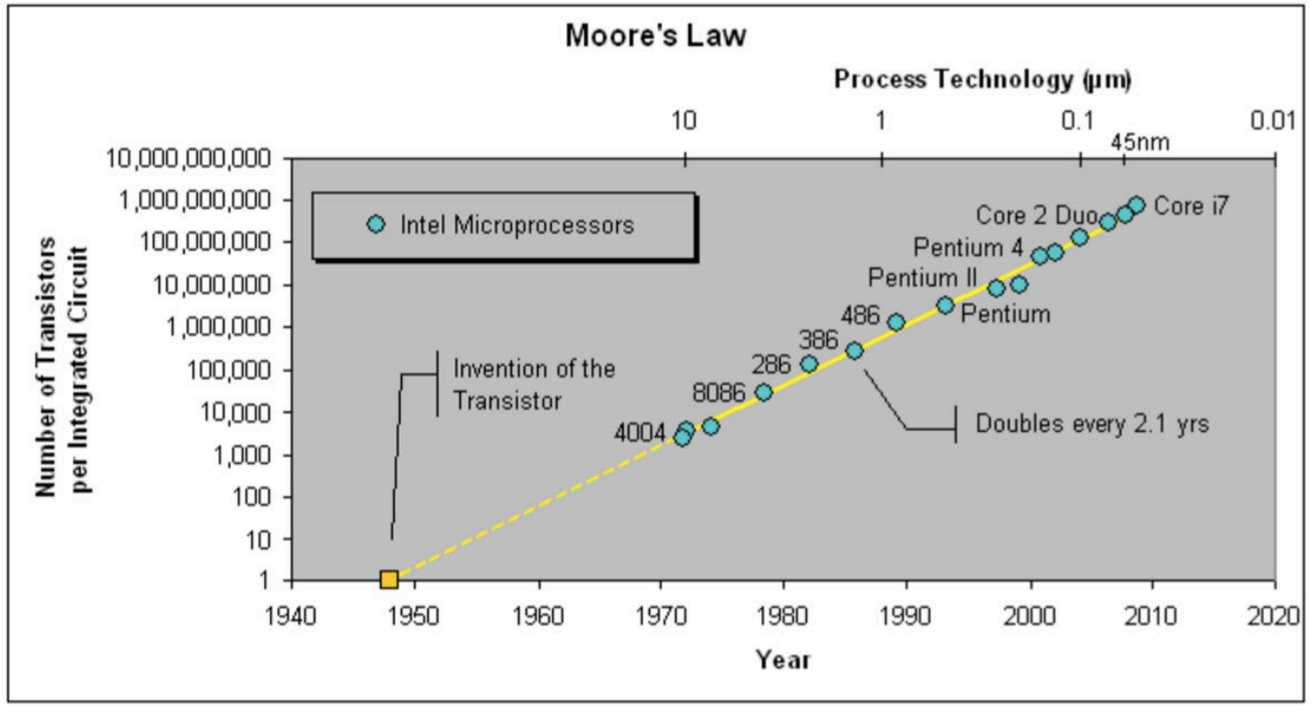
Designed by Freepik

Sure but, Why ?

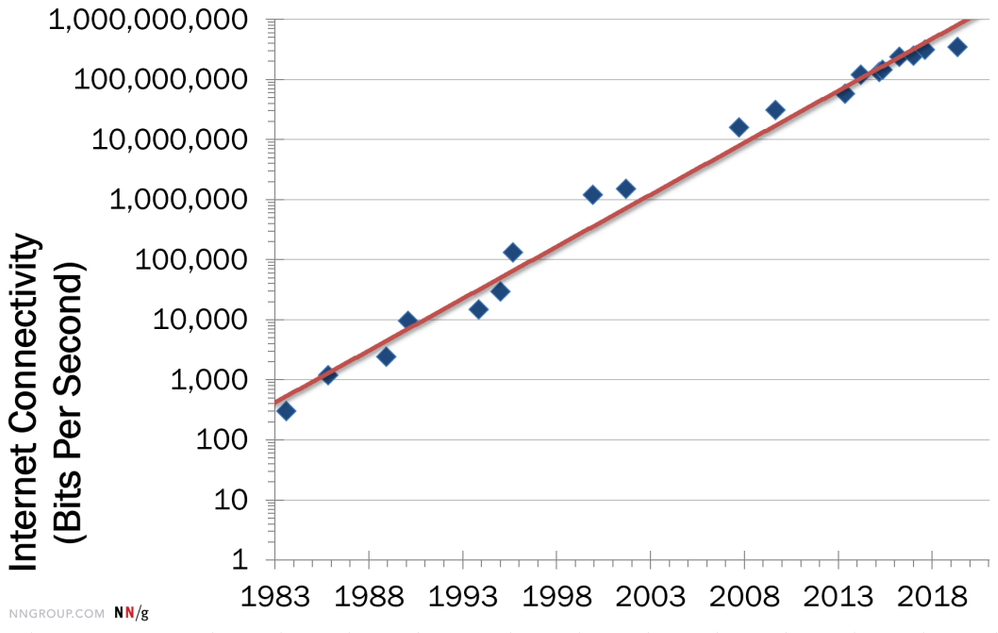
- **Why is more important than how**

Moore's, Nielsen's and Disk Space Law

- Computer power, speed of broadband and disk storage capability doubles every year
- The same computer power, speed of broadband and disk storage capability will cost half the current price in 2 years' time



Image/AnandTech



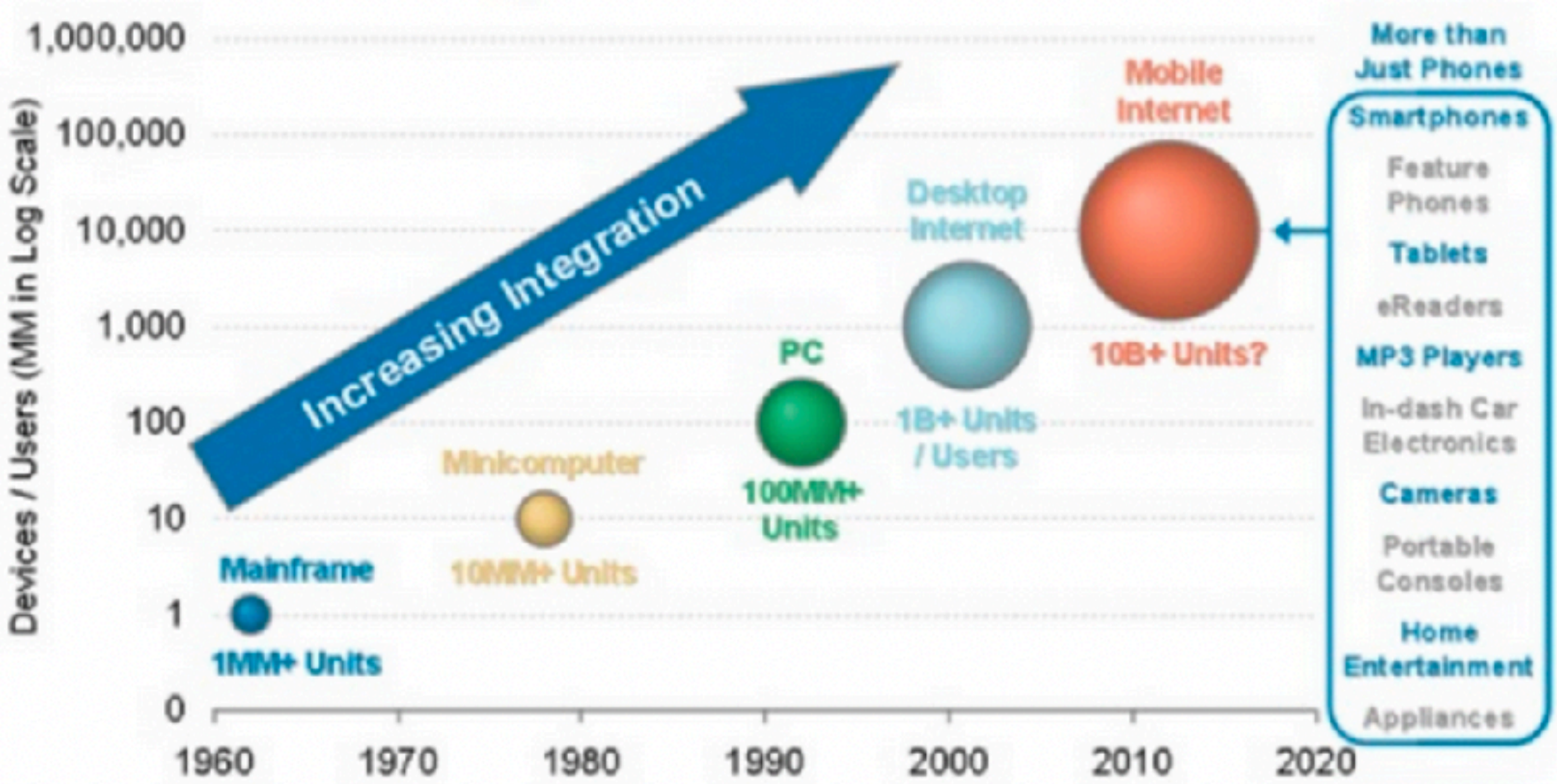
NNGGroup.com

Each New Computing Cycle = 10x > Installed Base than Previous Cycle

Exhibit 29

Each new computing cycle typically generates around 10x the installed base of the previous cycle

Devices or users in millions; logarithmic scale



@KPCB Source: Morgan Stanley Mobile Internet Report (12/09)

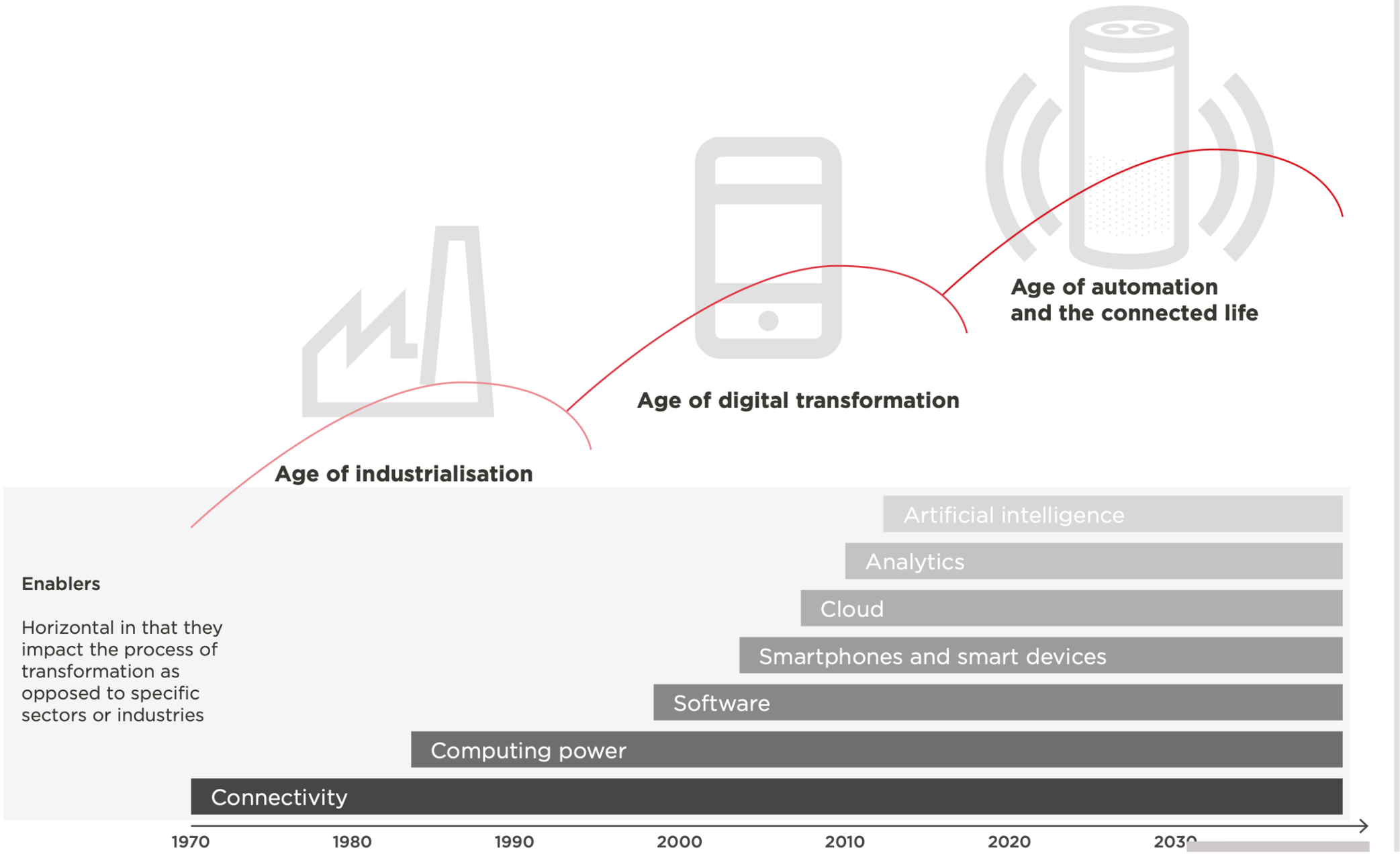
11

http://www.kpcb.com/internet-trends

The University Of Sheffield.

The shape of the things to come

Towards ubiquitous computing



Source: https://www.gsmaintelligence.com/research/?file=357f1541c77358e61787fac35259dc92&download

Ubiquitous computing

- Digital technologies assist a person 24/7
- Where we can understand their e.g. behaviour
- It is a world where we can:
  - Provide tailored advice based on what they are doing
  - Optimise urban environments based on human behaviour
- Two things will happen
  - The world as we know it will change radically
  - The type of expertise we need will change radically

Mobile devices

- The first mobile phones were two-ways radios
- Motorola created the first mobile phone in 1973
  - In 1983, the first mobile phones went on sale in the U.S, at almost $4,000 each

Mobile devices

1997 – Nokia 6110

1999 – BlackBerry 850

2000 – Nokia 3310

2002 - Samsung SGH-T100

2006 – Nokia N95

Mobile devices

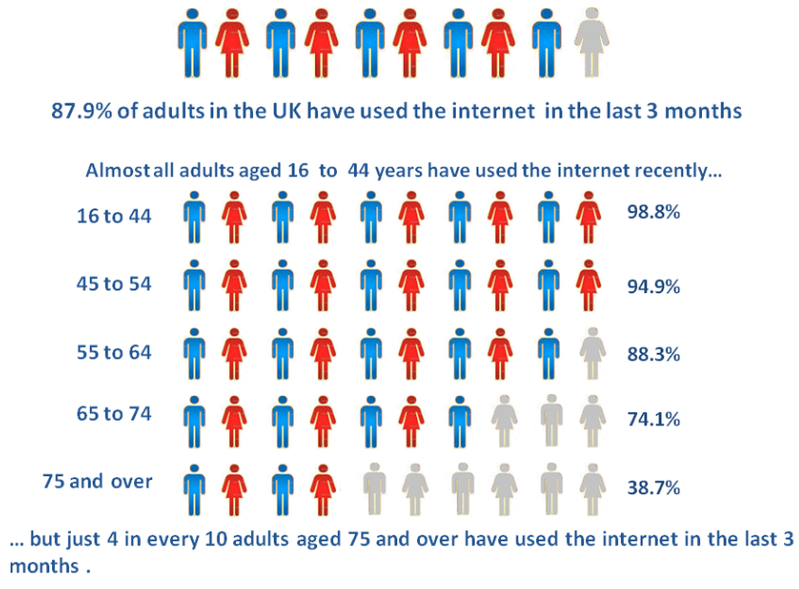
Source: https://www.slideshare.net/sqrajper/mobile-computing-24722802

Why is Mobile Computing Important

- Mobile phones are **more and more powerful**
- New services are **available everyday**
  - Some of which are not available on desktop computers
  - E.g. Uber
  - Relies on identifying user location to track the nearest taxi
- How do you do that precisely on a desktop ?

Internet connectivity

- Internet usage is widespread in the UK and worldwide
- According to the UK Office for National statistics
  - 87.9% of adults have used internet in the last 3 months
- There is still an age bias
  - Looking at <45 years old the percentage increases to 98.8%
  - >75 only 38.7%
- But this will gradually disappear



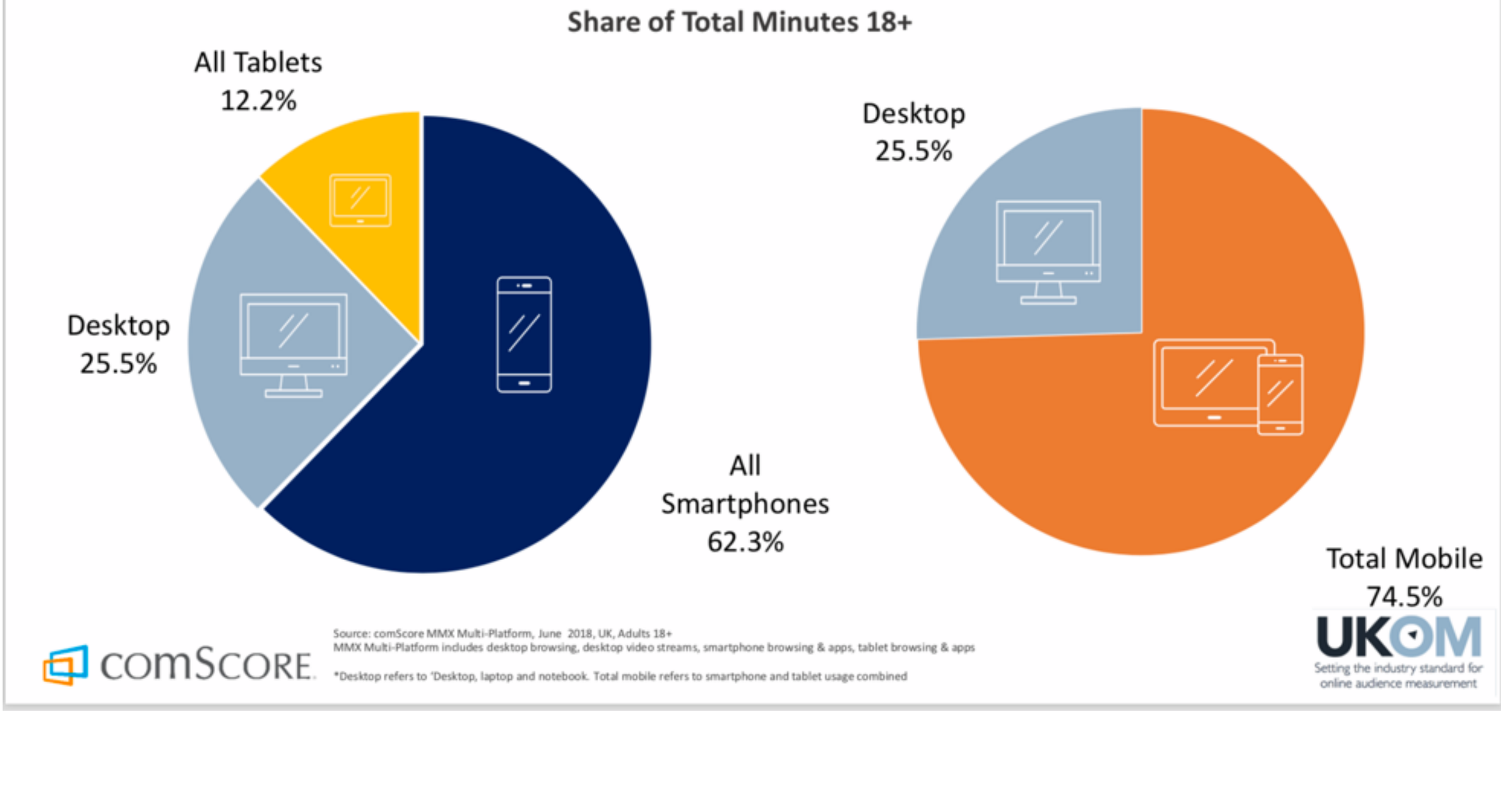
Source: Office for National Statistics

https://www.ons.gov.uk/businessindustryandtrade/itandinternetindustry/bulletins/internetusers/2016

The University Of Sheffield.

Share of Minutes by Platform

Smartphones now account for **62%** of all adult online minutes



comScore

Source: comScore MMX Multi-Platform, June 2018, UK, Adults 18+
MMX Multi-Platform includes desktop browsing, desktop video streams, smartphone browsing & apps, tablet browsing & apps

*Desktop refers to 'Desktop, laptop and notebook. Total mobile refers to smartphone and tablet usage combined

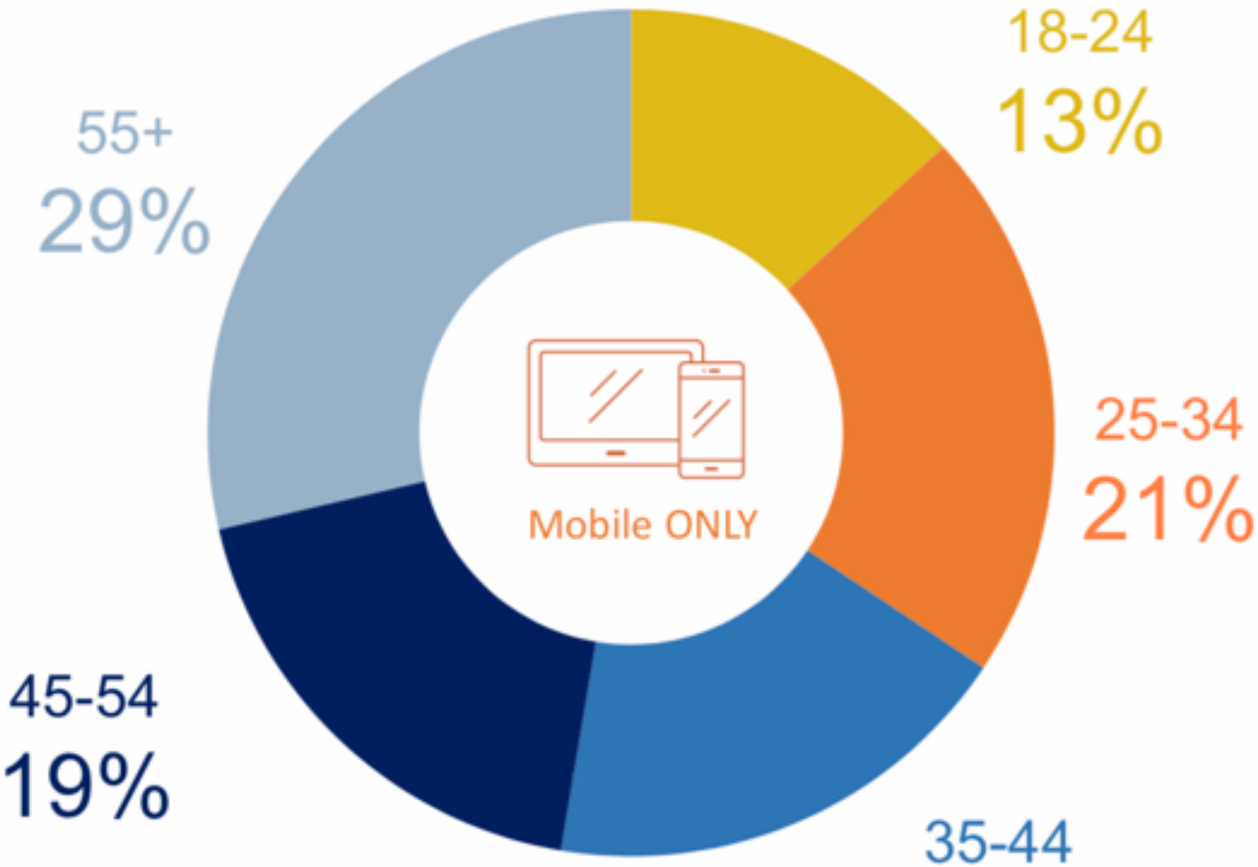
UKOM
Setting the industry standard for online audience measurement

The University Of Sheffield.

Mobile Only Audience Unique Visitors by Age

The 'mobile only' audience is split across **all age ranges**

Tablets drive up 55+ share of the Mobile Online Audience



comScore

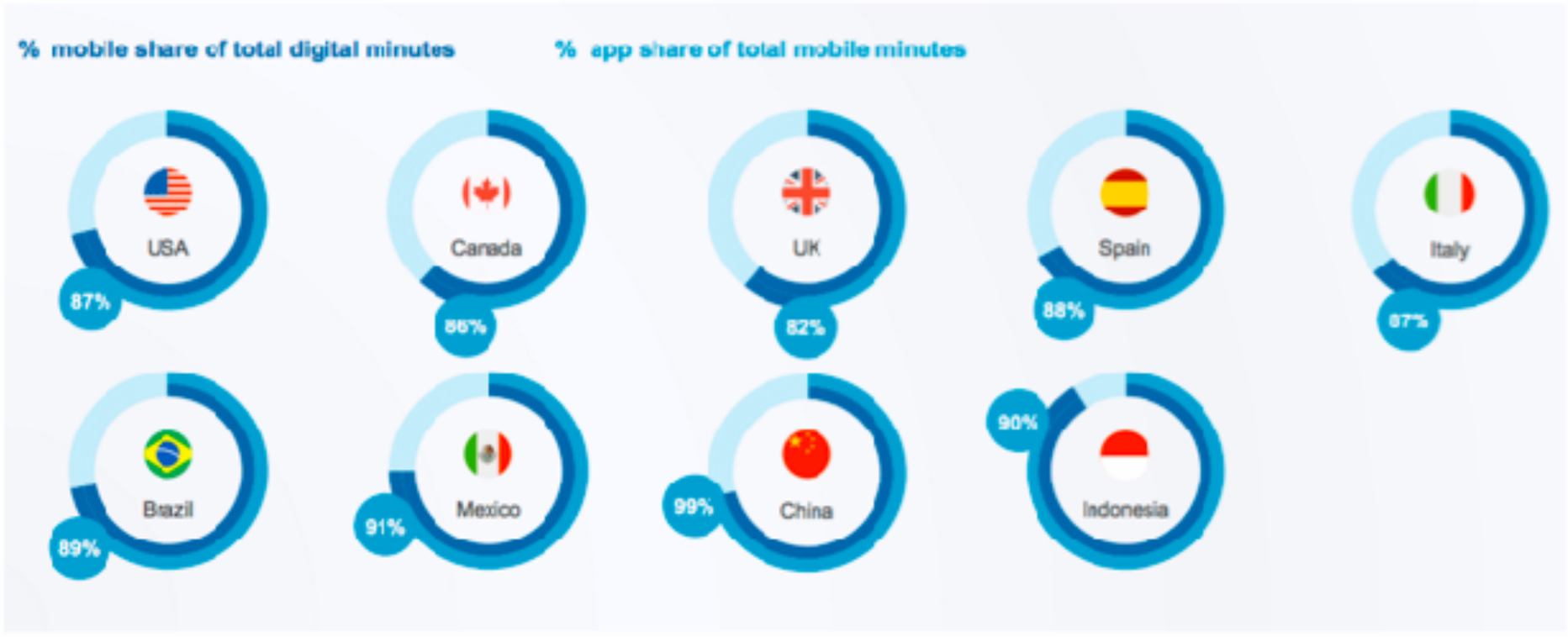
Source: comScore MMX Multi-Platform, June 2018, UK Adults 18+

MMX Multi-Platform includes desktop browsing, desktop video streams, smartphone browsing & apps, tablet browsing & apps

UKOM
Setting the industry standard for online audience measurement

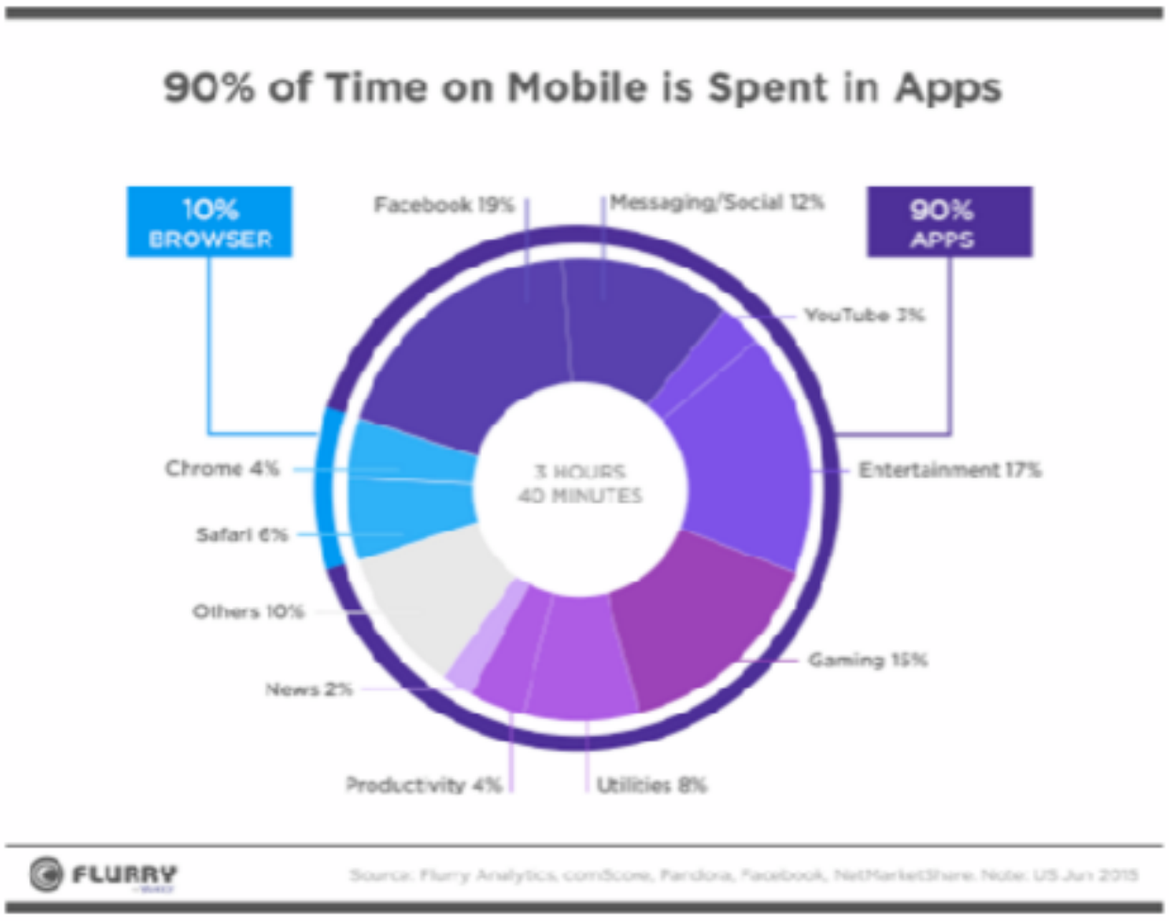
The App age

- People are increasingly using apps instead of mobile web browsers
- In the UK, 82% of mobile minutes are app usage, In China 99%



App vs. Mobile Browsing

- Mobile apps are usage is driven by social networks, messaging, consumption of news and video



Apps account for over 90% of internet time on smartphones and 77% of internet time on tablets. Nearly half of app time occurs in an individual's top app, and 90% in the top five.

https://www.emarketer.com/content/mobile-time-spen

App vs. Mobile Browsing

- Mobile apps are easily accessible
- Can be launched faster
- Have a specific reason to be used
- Have a greater degree of functionalities
- Integrate with other apps
- Integrate with device capabilities •i.e. GPS, camera etc

.

Mobile Browsing

53% of users will abandon a site if it takes longer than 3 seconds to load!

Once loaded, users expect them to be fast—no janky scrolling or slow-to-respond interfaces

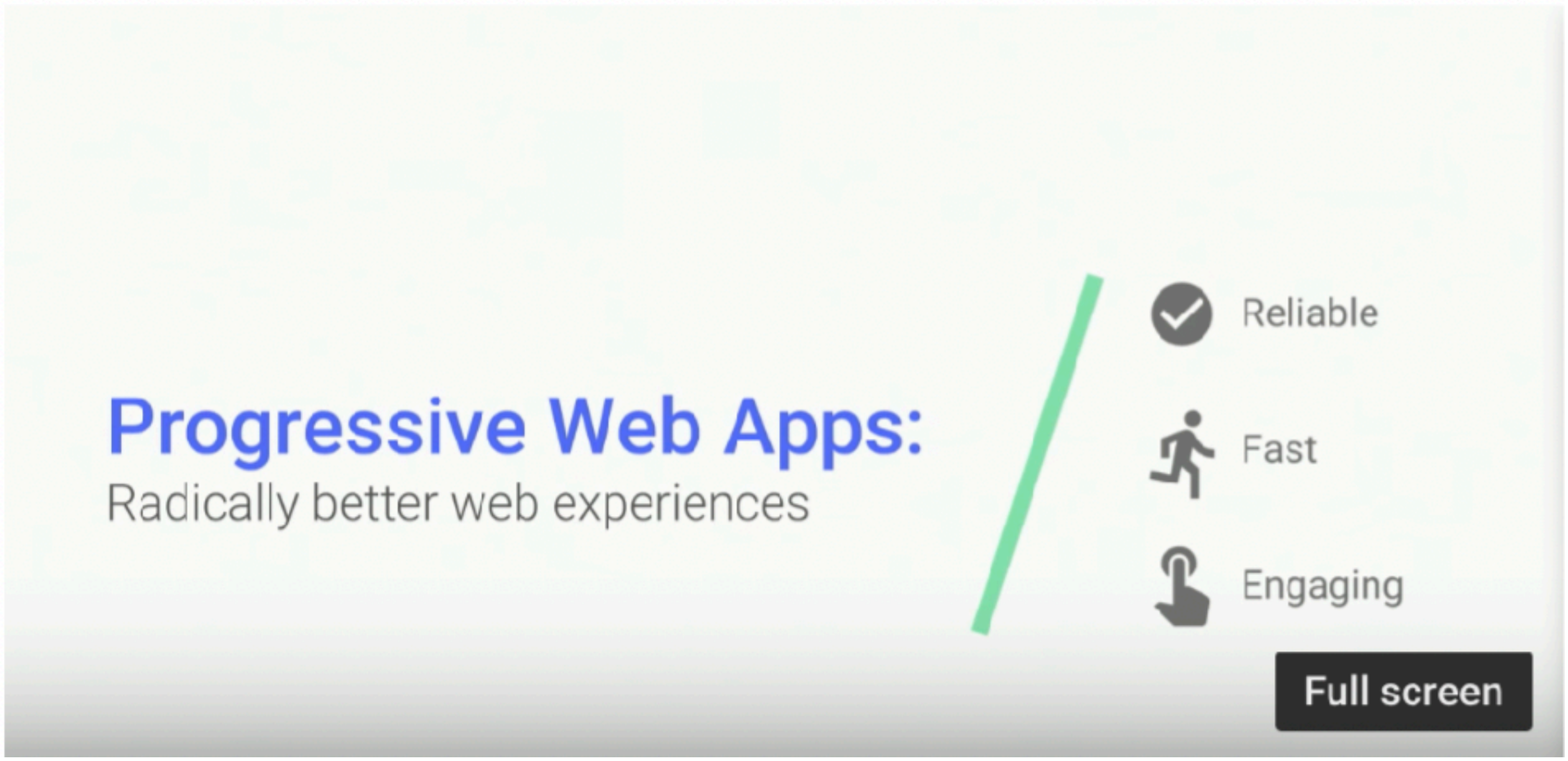
The real advantage of apps is the ability to provide a perfect offline experience

- Content may not be updated if offline but apps can take opportunistic strategies
  - to download content when online
  - to store content for off line viewing
- allow push notifications even when the user is using a different app

.

Progressive Web Apps

- Are websites that provide a native app experience without downloading an app
- they use web service workers
.



PWAs

Progressive - Work for every user, regardless of browser choice because they're built with progressive enhancement as a core tenet.

Responsive - Fit any form factor: desktop, mobile, tablet, or forms yet to emerge.

Connectivity independent - Service workers allow work offline, or on low quality networks.

App-like - Feel like an app to the user with app-style interactions and navigation. Fresh - Always up-to-date thanks to the service worker update process.

Safe - Served via HTTPS to prevent snooping and ensure content hasn't been tampered with.

Discoverable - Are identifiable as "applications" thanks to W3C manifests[6] and service worker registration scope allowing search engines to find them.

Re-engageable - Make re-engagement easy through features like push notifications.

Installable - Allow users to "keep" apps they find most useful on their home screen without the hassle of an app store.

Linkable - Easily shared via a URL and do not require complex installation

Wearables

- A product that you wear somewhere on your body, e.g.
  - Fitness-tracking bands
  - Smartwatches
  - Smartglasses

- Normally use low-power, low-cost sensors
- Allow to track data with very little battery drain

Wearables

- Gartner predicts sales of 310.4M wearable devices worldwide this year
- $30.5BN in revenues
- of which $9.3BN from the smartwatch category

Table 1: Forecast for Wearable Devices Worldwide 2016-2018 and 2021 (Millions of Units)

Device	2016	2017	2018	2021
Smartwatch	34.80	41.50	48.20	80.96
Head-mounted display	16.09	22.01	28.28	67.17
Body-worn camera	0.17	1.05	1.59	5.62
Bluetooth headset	128.50	150.00	168.00	206.00
Wristband	34.97	44.10	48.84	63.86
Sports watch	21.23	21.43	21.65	22.31
Other fitness monitor	55.46	55.7	56.23	58.73
Total	**265.88**	**310.37**	**347.53**	**504.65**

Source: Gartner (August 2017)

Lecture Overview

- Part 1: What is Mobile Computing?
  - A brief history
  - Current trends
- **Part 2:What is Android?**
  - Architecture
  - App components
  - Entry points and interacting

Question:

Do I need an Android phone to participate in the module?

Answer: no: you will use the emulator
(you can keep your iPhone if you really must) 

Android

- Android apps are written in Java **and/or** Kotlin
  - For simplicity we will just see Java
- Android SDK compiles your code along with any data and resource files
  - into an APK, an Android package
    - an archive file with an .apk suffix.
    - that contains all the contents of an Android app
    - that is used to install the app
- Each Android app lives in its own security sandbox

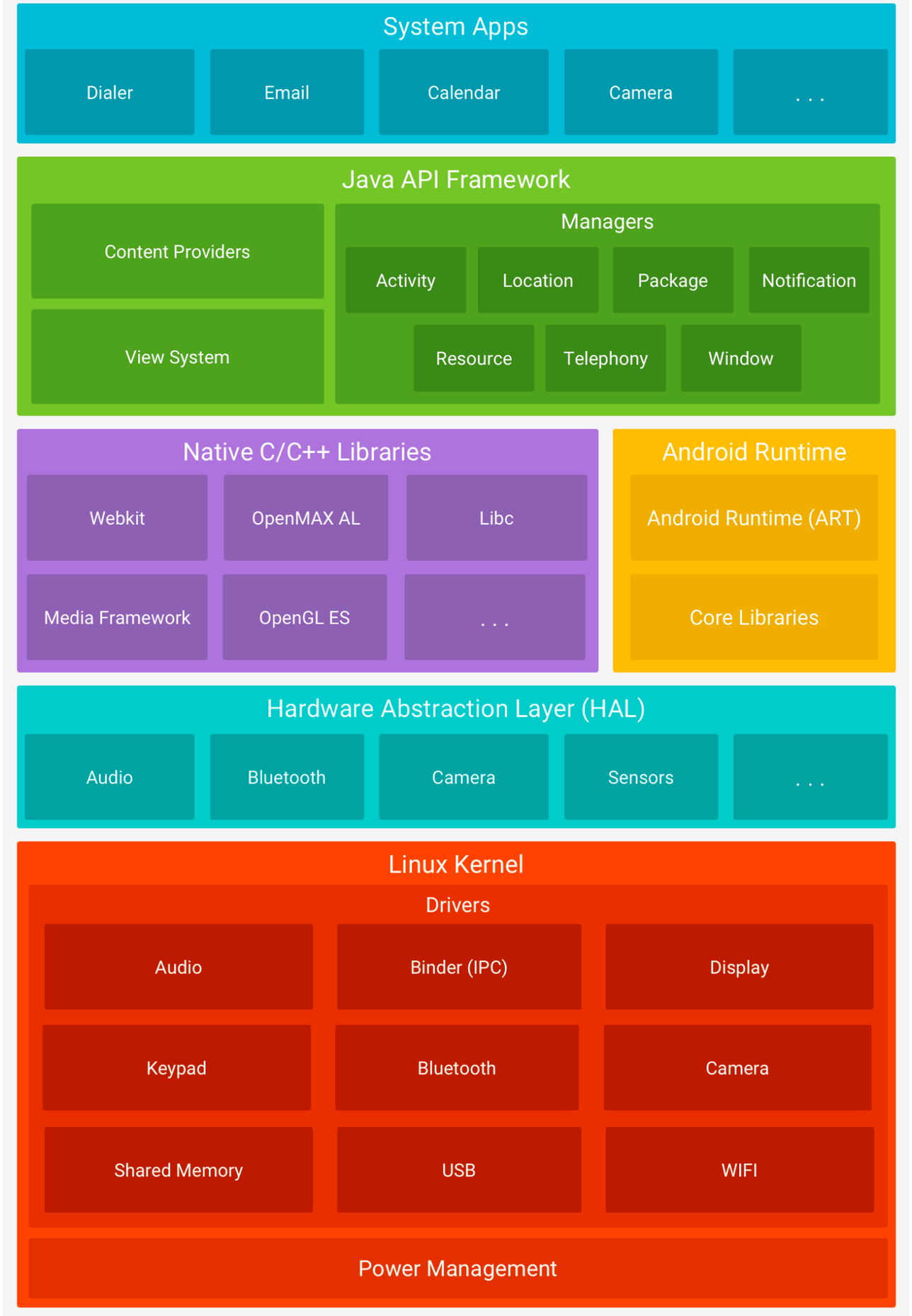
https://developer.android.com/guide/components/fundamentals.html#Components

The University Of Sheffield.

Architecture

The Android software stack

https://developer.android.com/guide/platform/index.html



A Secure Linux System

- Android OS is a **multi-user Linux system**
  - Each app is a different user
- The system assigns each app a **unique Linux user ID**
  - unknown to the app
- The system sets permissions for all the files in an app
  - so that only the user ID assigned to that app can access them
- Each process has its own **virtual machine (VM)**, so an app's code runs in isolation from other apps
- By default, every app runs in its own Linux process

Execution

- The Android system starts the process when any of the app's components need to be executed
  - It then shuts it down
    - when it's no longer needed
    - when the system must recover memory for other apps
- Android is based on **the principle of least privilege**:
  - each app, by default, has access **only** to the components needed to do its work
  - This creates a very secure environment in which an app cannot access parts of the system for which it is not given permission

Permissions

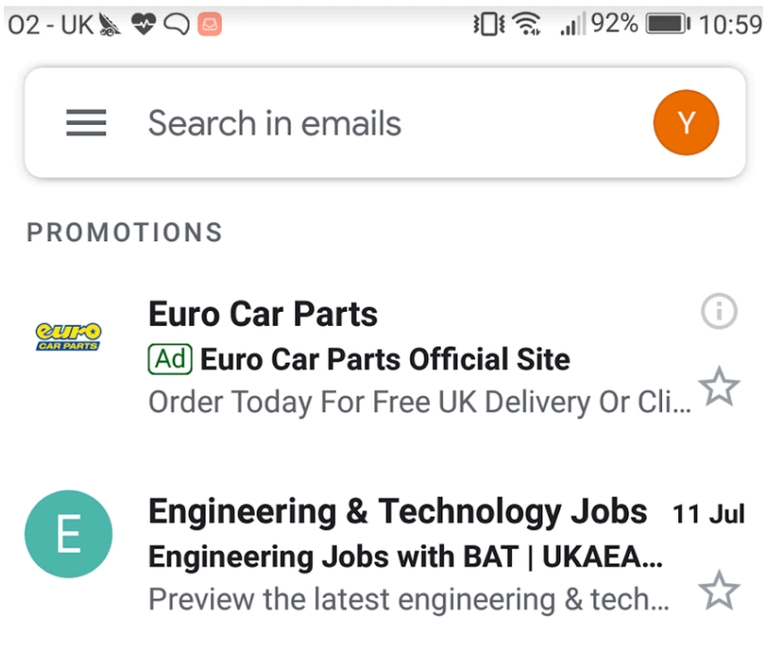
- There are ways for an app to share data with other apps and for an app to access system services:
- An app can **request permission** to access device data
  - such as the user's contacts, SMS messages, the SD card, camera, and Bluetooth
  - The user has to explicitly grant these permissions
    - Android M (6.0 and higher) has introduced two types of permissions:
      - Dangerous permissions (user has to accept)
      - Other permissions (user is not asked)

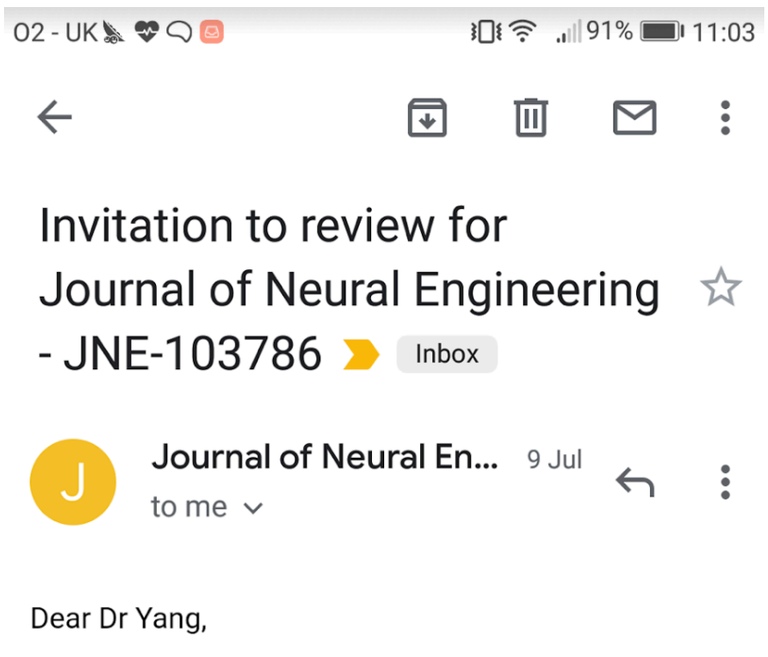
App Components

- **4 types**
  - Activities
  - Services
  - Broadcast receivers
  - Content providers
- Each type serves a **distinct purpose** and has **a distinct lifecycle**
  - Defining how the component is created and destroyed

Activity

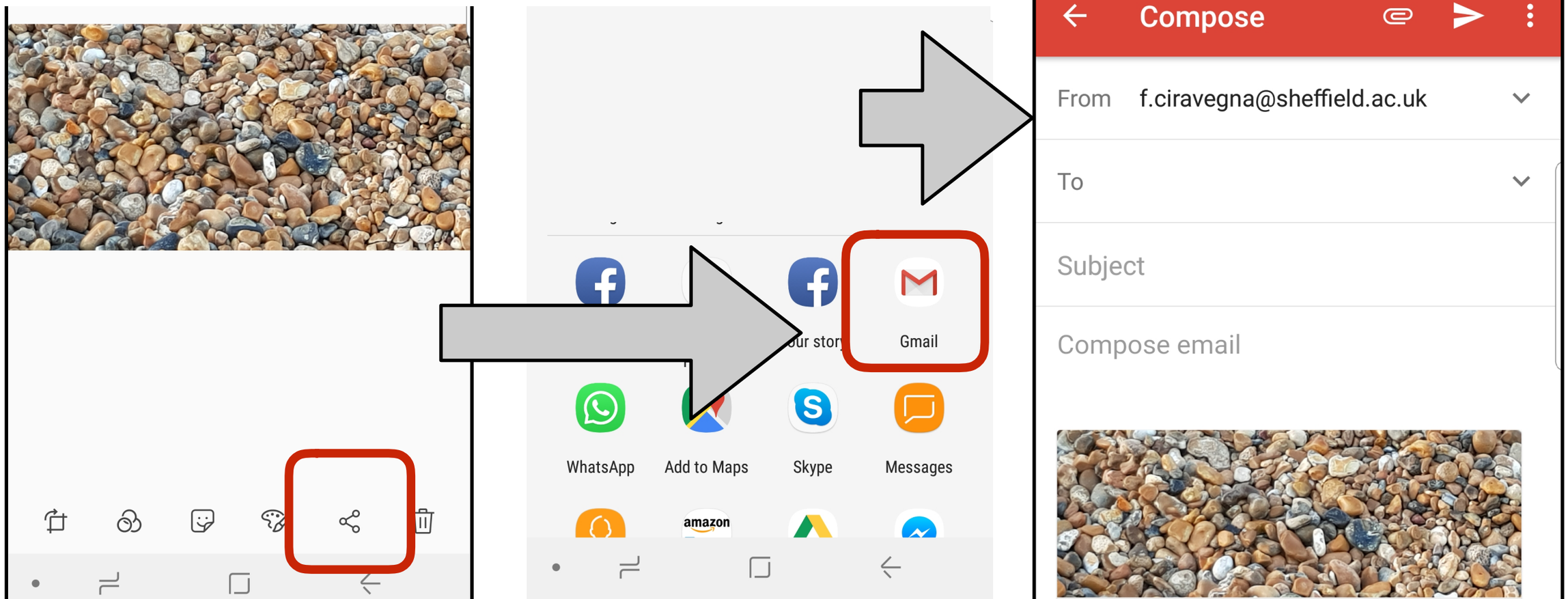
- The entry point for user interaction
- It represents a single screen with a user interface
  - For example, an email app might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails





Activities

- Activities work together to form a cohesive user experience in an app,
  - each one is independent of the others
  - A different app can start **any** one of these activities if the email app allows it
  - For example in order to share, a camera app can start the activity in the email app that composes new mail to allow the user to share a picture
    - in the share option of the camera



Services

- A general-purpose entry point for keeping an app running in the background
  - Typically used to perform long-running operations or to perform work for remote processes
    - never ending services
    - services that stop after a task is performed
- A service does **not provide a user interface**
  - You must implement an Activity for interacting with users
- Example:
  - a service might play music in the background while the user is in a different app, or it might fetch data over the network without blocking user interaction with an activity

Services (2)

- Services are typically started from another component
  - e.g. an activity or a broadcast receiver
  - so to perform operations in the background
  - It is possible to **bind the service to another** component to interact with it

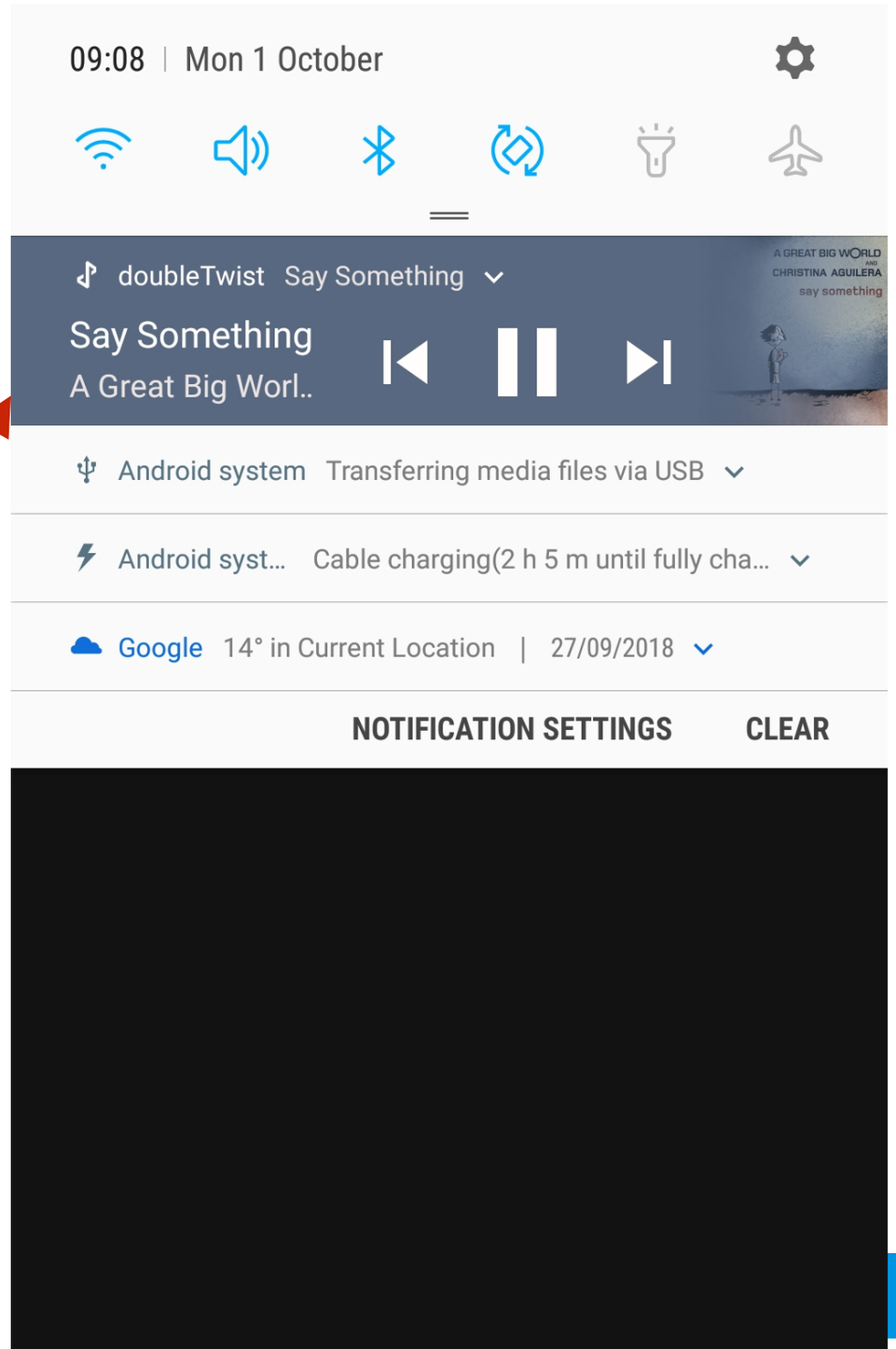
Services (3)

- Foreground services and background services
  - **Foreground services**:
    - the user is fully aware of them and **their experience would be reduced if stopped or delayed**
    - e.g. a music player plays music when the user is doing something else
    - they cannot be stopped or optimised
    - Permanent notification on lock screen is required since Android M
  - **Background services**: perform hidden operations the user is largely unaware of
    - e.g. sending data to a server
    - they can be stopped and restarted if memory is needed
      - their way of working may be "optimised"
  - **Middle ground services:**
    - e.g. a step counter service is running in the background and the user is not fully aware of it. nonetheless it cannot be stopped or optimised (these are becoming unacceptable in Android)

Foreground service example

Pull down

Permanent Notification



Broadcast Receivers

- A component that enables the system to deliver events to the app outside of a regular user flow,
  - To respond to system-wide or app-wide broadcast announcements
- Another entry into the app
- The system can deliver broadcasts even to apps that aren't currently running
- For example, an app can schedule an alarm to post a notification to tell the user about an upcoming event
    - By delivering that alarm through a BroadcastReceiver, there is no need for the app to remain running until the alarm goes off

BR (2)

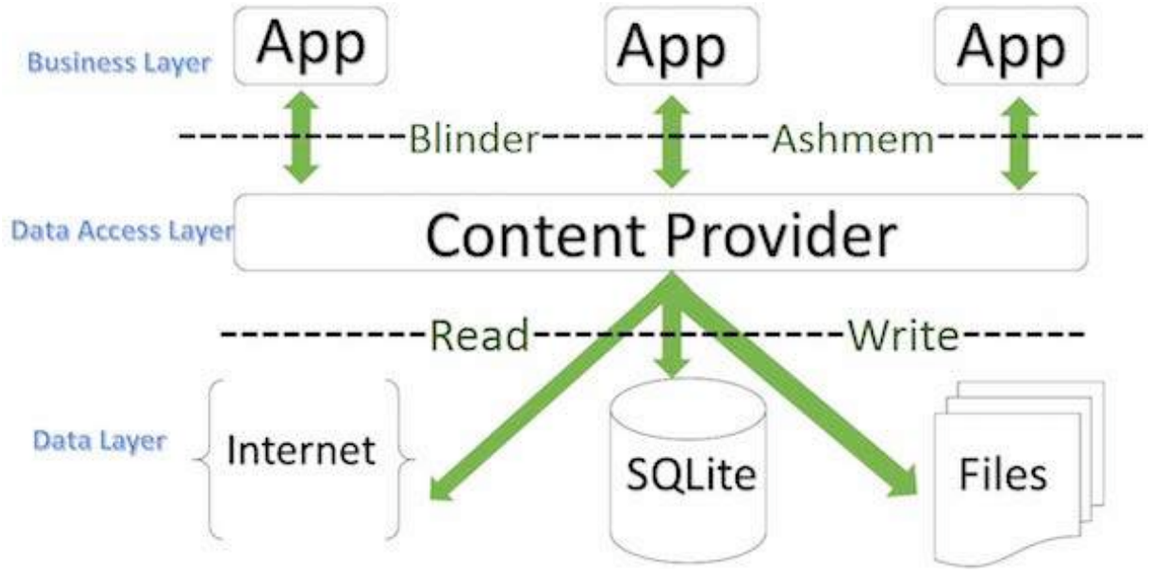
- Many broadcasts originate from the system
  - e.g. screen off, screen on, battery low, etc.
- Apps can initiate broadcasts
  - e.g. some data has been downloaded to the device and is available for them to use
  - it can also be used to communicate between single app's components
- Broadcast receivers don't display a user interface
  - They may create a status bar notification to alert the user when a broadcast event occurs
- Many system BR are now being disallowed because of abuse by apps
  - e.g. many apps used screen_on needlessly to fire up heavy operations
    - battery usage was very high

Content Providers

- Manage a shared set of app data that you can store
  - in the file system,
  - in a SQLite database,
  - on the web,
  - or on any other persistent storage location
- Other apps can query or modify the data if the content provider allows it.
  - For example, the Android system provides a content provider that manages the user's contact information.

Content Providers

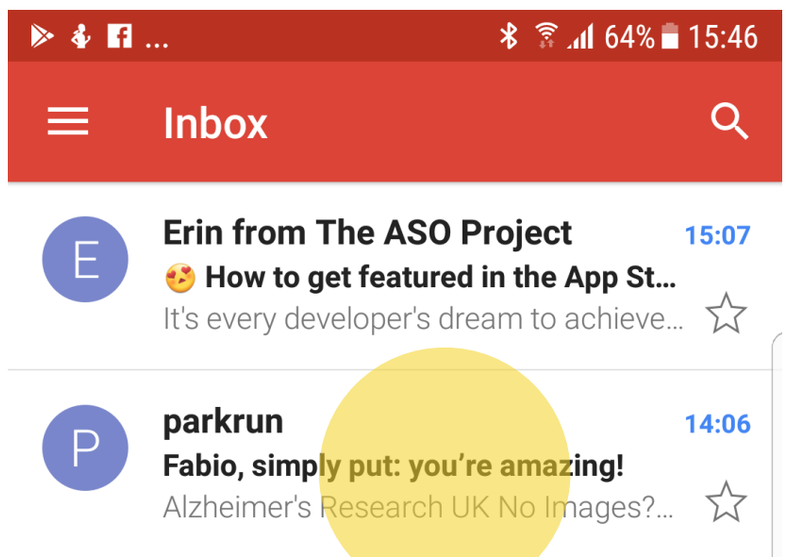
- Manage a shared set of app data that you can store
  - in the file system,
  - in a SQLite database,
  - on the web,
  - or on any other persistent storage location

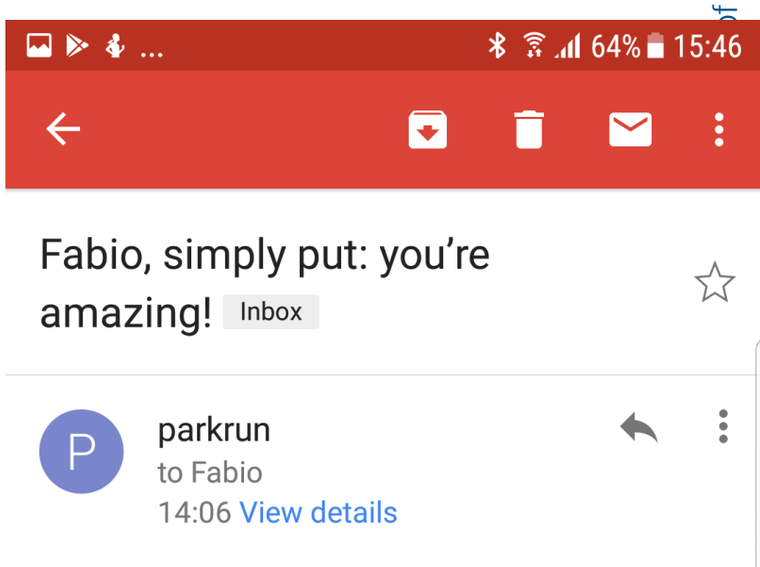


- Content providers allow both **public and private** data reading and writing

There is no main program

- Android has **no Java/Kotlin main program**
- Entry points are the components
  - A component (e.g. activity) can be opened by the user by tapping an icon on the app list
  - A component can be launched by another component
    - e.g. the email list will open the message display activity when we tap on a message





Starting an activity

- So how does touching the icon start an activity?
  - Using a Broadcast receiver
  - Your app must declare the entry point for pressing the icon:
    - later we will see how this is done. In short: the **`AndroidManifest.xml`** file declares the activities and the way to invoke them using BRs. One of them will respond to the **`launcher`** BR

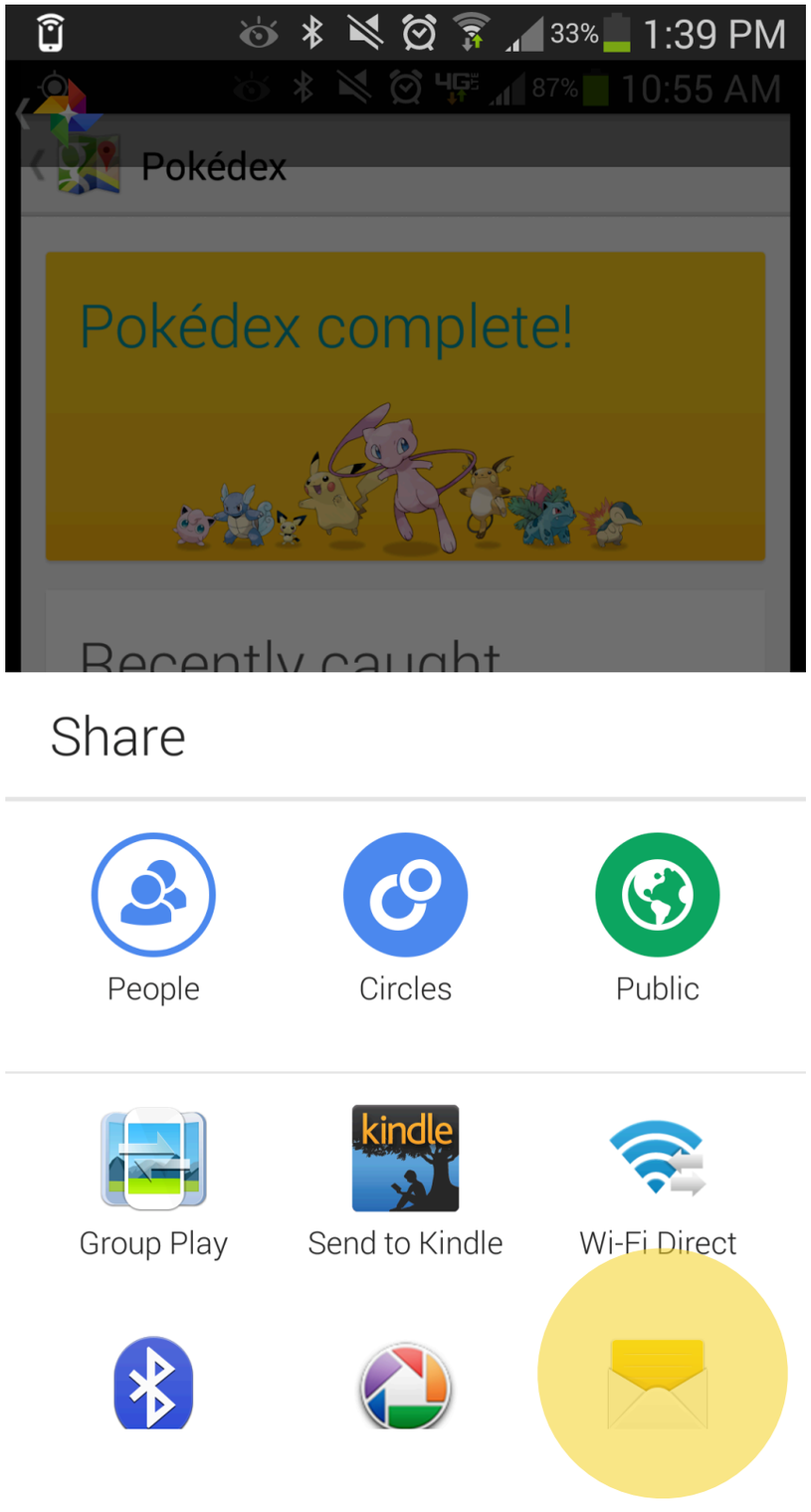
```
<activity
    android:name=".SplashActivity"
    android:screenOrientation="portrait"
    android:theme="@style/FullScreen">
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
```

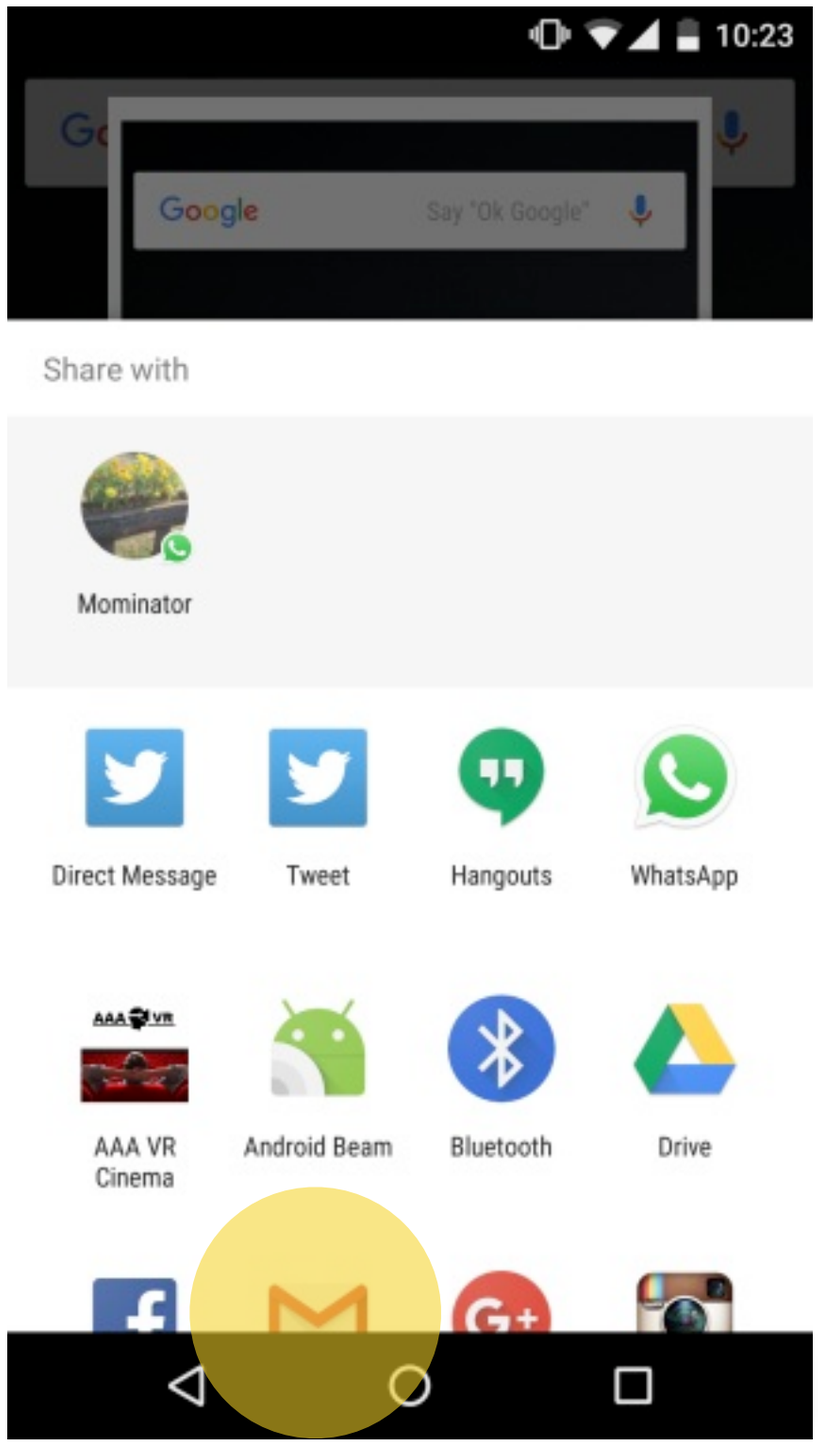
Calling other apps

- Any app can start another app's component if declared as public
  - For example, if you want the user to capture a photo with the device camera, there's probably another app that does that and your app can use it instead of developing an activity to capture a photo yourself.
- You don't need to incorporate or even link to the code from the camera app.
    - You can simply start the activity in the camera app that captures a photo.
- When complete, the photo is even returned to your app so you can use it. To the user, it seems as if the camera is actually a part of your app.

Sharing

When you share by email, you call the email compose activity directly





App Data Space

- When the system starts a component, it starts the process for that app if it's not already running and instantiates the classes needed for the component
- The calling app has no access to the file system of the called app
  - the new app is run in its own VM
  - So no app can interfere with other app's data

How to activate another component

- Android runs **each app in a separate process with file permissions** that restrict access to other apps
- Your app **cannot directly activate a component** from its own space or from another app
- It will deliver a message to the **system** that specifies your ***intent*** to start a particular component
  - The system then activates the component for you
  - It is called an intent because it is a request (it can be denied, in principle)

Activating components

- Activities, services, and broadcast receivers are activated by an *asynchronous message* called an ***Intent***
- Intents bind individual components to each other at runtime
  - You can think of them as the messengers that request an action from other components
  - internal to this app or internal to another app

Creating an Intent

- An intent is created with an Intent object,
  - it defines a message to activate either a specific component (explicit intent) or a specific type of component (implicit intent)
- For activities and services, an intent defines the action to perform and may specify the URI of the data to act on
  - For example, an intent might request to open a web page
  - You can also start an activity to receive a result when the activity completes
    - e.g. to pick a date from a calendar

Calling an Intent

- To start an activity

```
Intent intent = new Intent(this, MyActivity.class);
startActivity(intent)
```

  - or `startActivityForResult(intent)`

- To schedule actions:
  - With Android 5.0+ (API level 21) use JobScheduler class
  - For earlier Android versions
    - start a service by passing an Intent to startService()
    - You can bind to the service by passing an Intent to bindService()

Start Broadcast Receivers

- Pass an Intent to methods such as
  - `sendBroadcast()`
  - `sendOrderedBroadcast()`
  - `sendStickyBroadcast()`

- Example:

```
Intent broadcastIntent =
            new Intent("uk.ac.shef.oak.myPackage.myFlag");
sendBroadcast(broadcastIntent);
```

Summary

- Part 1: What is Mobile Computing?
  - A brief history
  - Current trends
- Part 2:What is Android?
  - Architecture
  - App components
  - Entry points and interacting
- Next - Lab tutorial:
  - Intro to this module