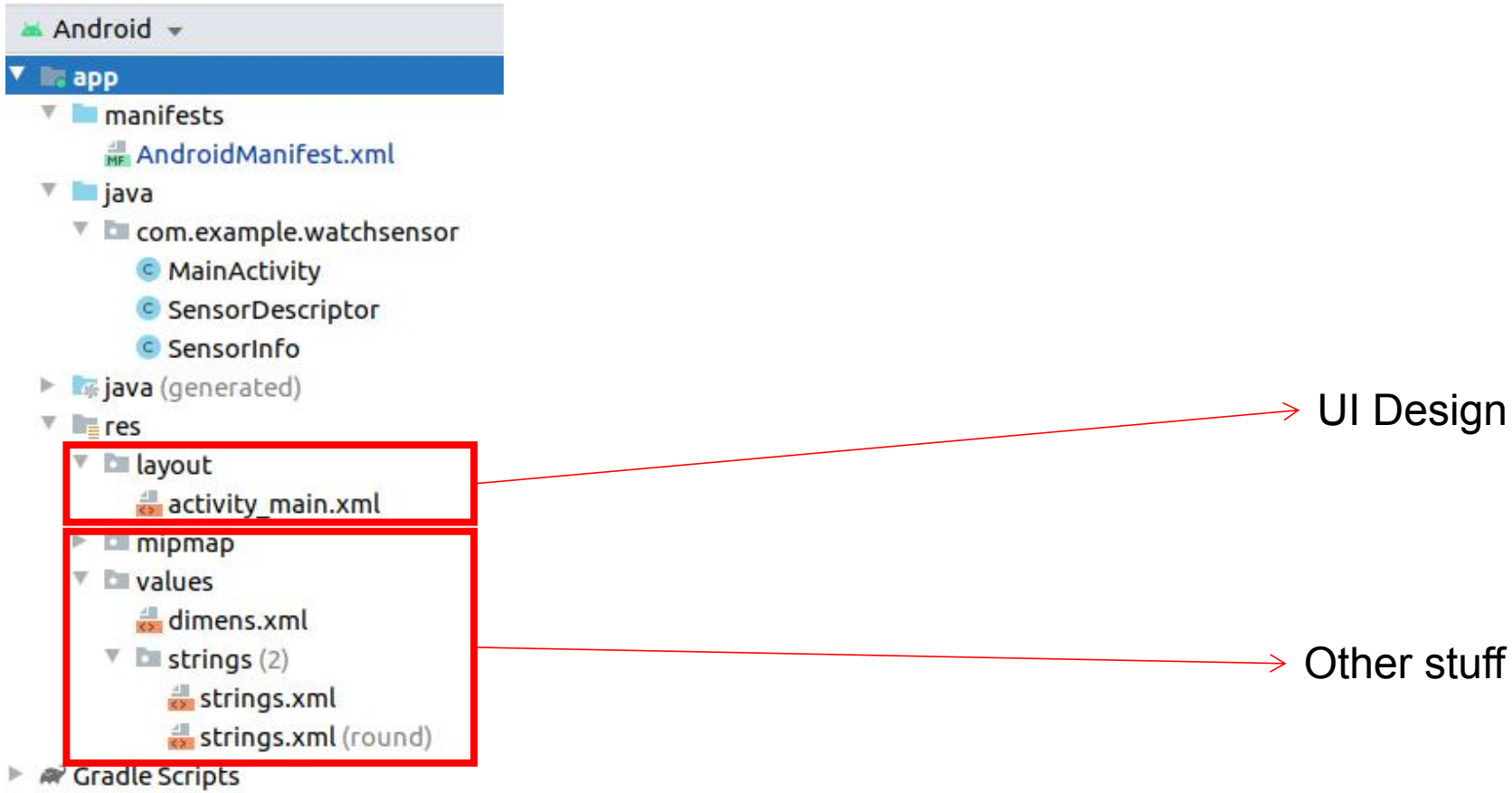
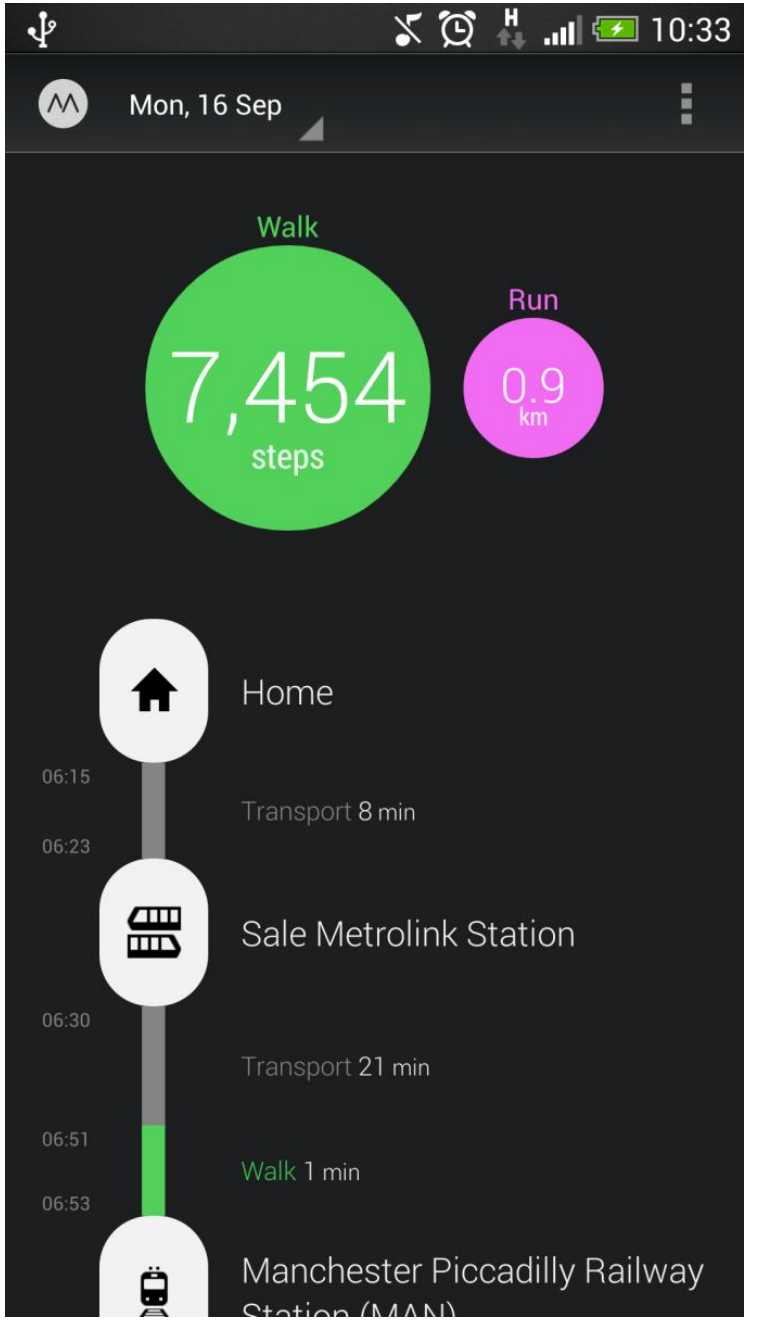
Mobile Application Architecture (Continued)

Designing UIs in Android


Android
app
manifests
AndroidManifest.xml
java
com.example.watchsensor
MainActivity
SensorDescriptor
SensorInfo
java (generated)
res
layout
activity_main.xml
mipmap
values
dimens.xml
strings (2)
strings.xml
strings.xml (round)
Gradle Scripts
UI Design
Other stuff

Why use XML?

- Question: Can we design UI using Java, without XML?
- Example: Shapes, colors can be changed in XML file without changing Java program
- UI designed using either:
  - Drag-and drop graphical (WYSIWYG) tool or
  - Programming Extensible Markup Language (XML)
- XML: Markup language, both human-readable and machine-readable‘’
- Purpose: separate UI from Logic



Implement UI in xml

- Example 1

```
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/textview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:text="@string/hello"/>
```

Implement UI in Java

```
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    TextView tv = new TextView(this);
    tv.setText("hello");
    LinearLayout ll = new LinearLayout(this);
    ll.addView(tv);
    setContentView(ll);
}
```

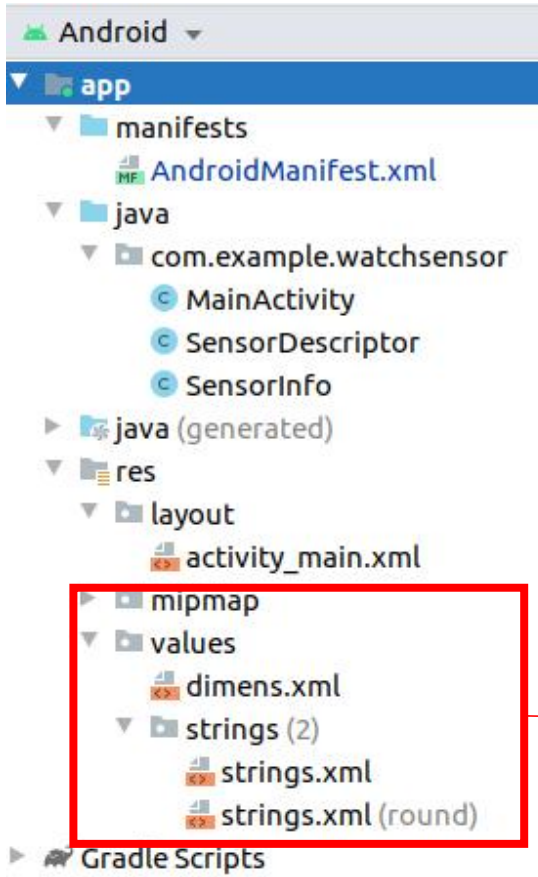
- The same UI can be implemented using java alone vs. java+xml
- Why is xml used and recommended?

Benefits of using xml:

- Separation of logic from presentation.
  - This does help on larger projects when you need to refactor some code.
- The structure of XML correlates nicely to the structure of a user interface, i.e., a tree like structure.

Other stuff

- Typically referred to as “assets,” anything that isn’t code is placed in the res/ folder
- String
- Music
- Images
- Some static data files


Android
app
manifests
AndroidManifest.xml
java
com.example.watchsensor
MainActivity
SensorDescriptor
SensorInfo
java (generated)
res
layout
activity_main.xml
mipmap
values
dimens.xml
strings (2)
strings.xml
strings.xml (round)
Gradle Scripts

Other stuff

Declaring Strings in Strings.xml

Can declare all strings in strings.xml

String declaration in strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">EmPubLite</string>
    <string name="hello_world">Hello world!</string>

</resources>
```

Reference string in a main_layout.xml files

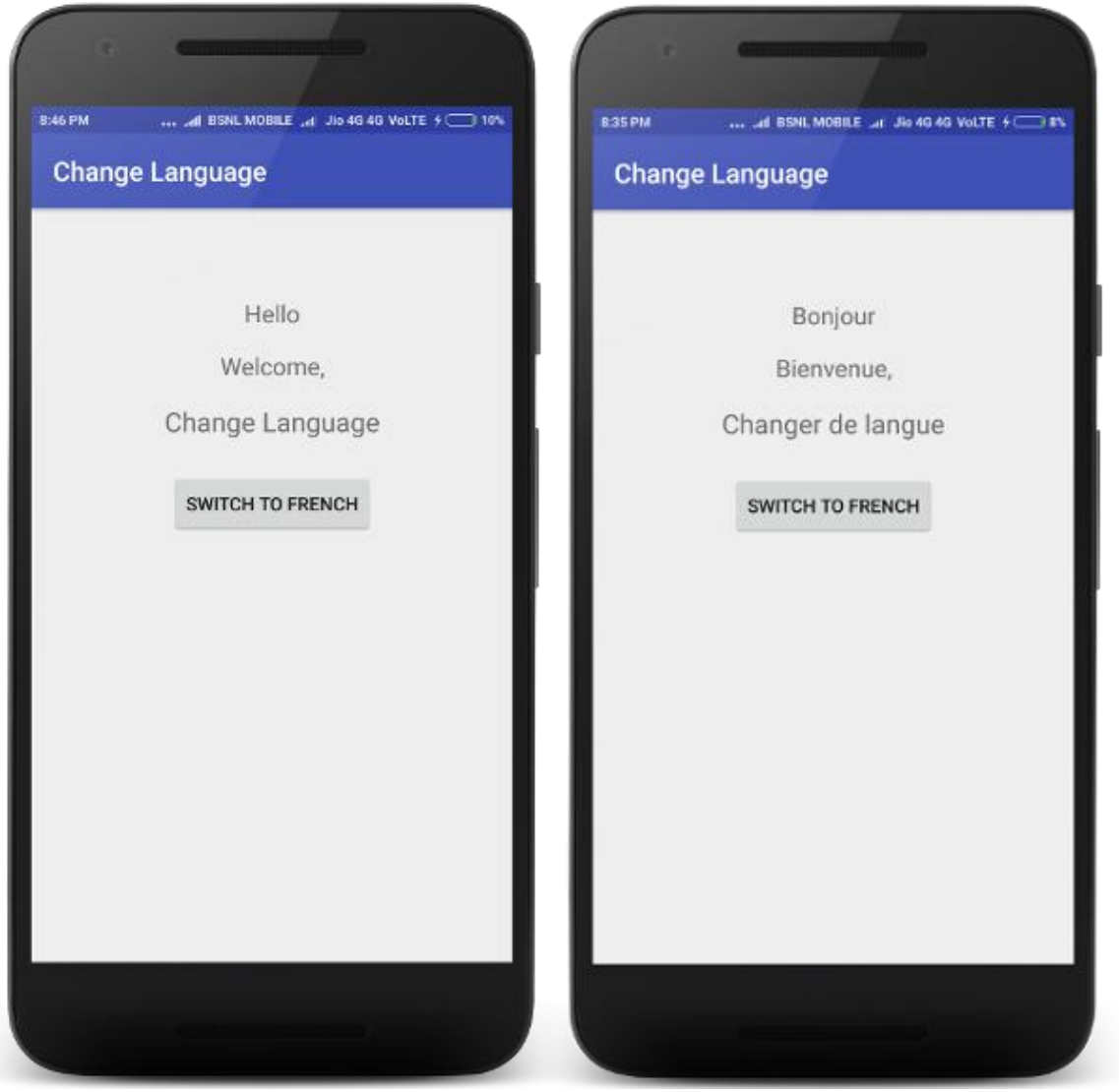
```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".EmPubLiteActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text="@string/hello_world"/>

</RelativeLayout>
```

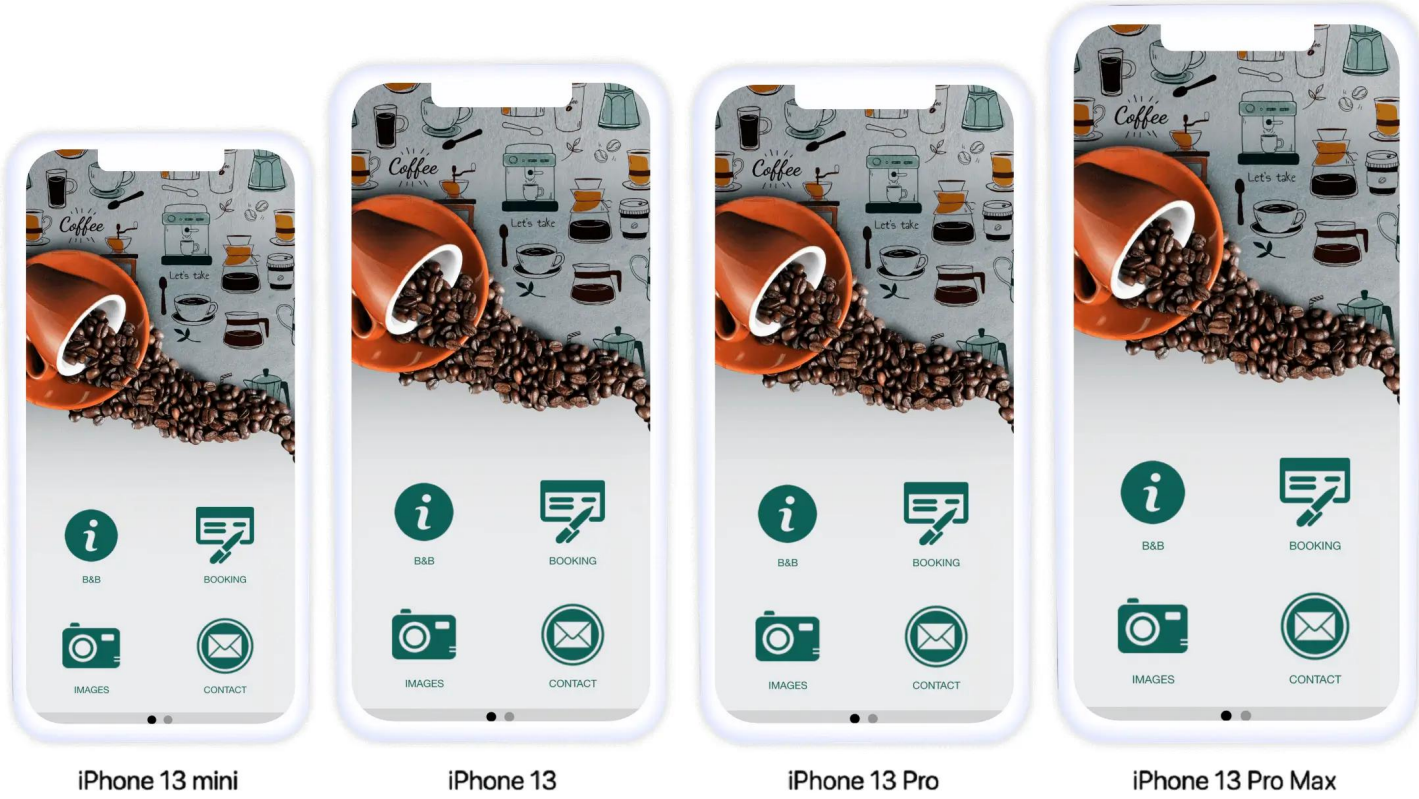
Why bother to use an additional string.xml file?

- Think about the scenarios:
  - translate to a new language



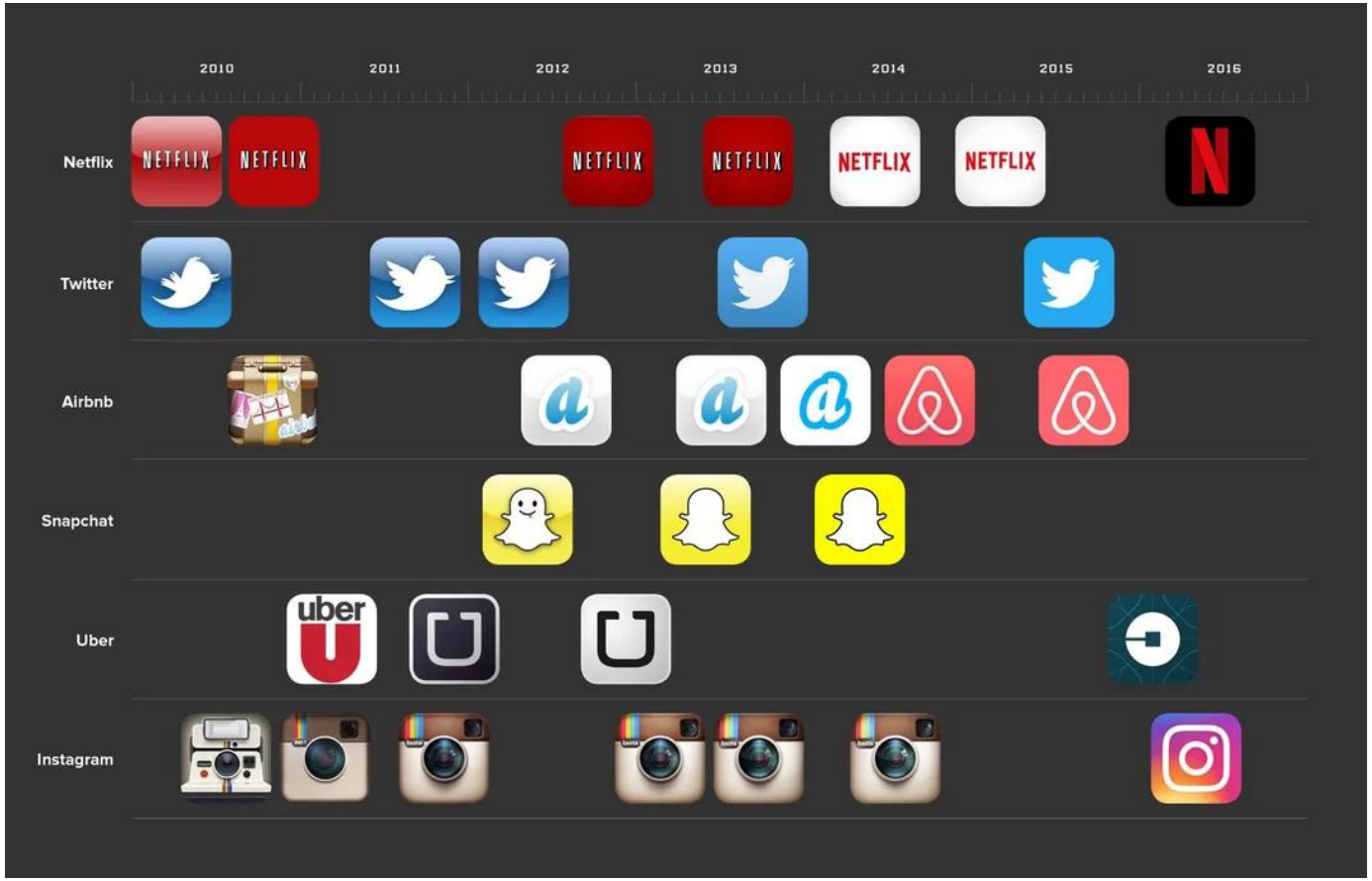
Why dimens.xml? The same thing can be implemented in Java.

- Think about the scenarios:
  - adapt app to different screen sizes



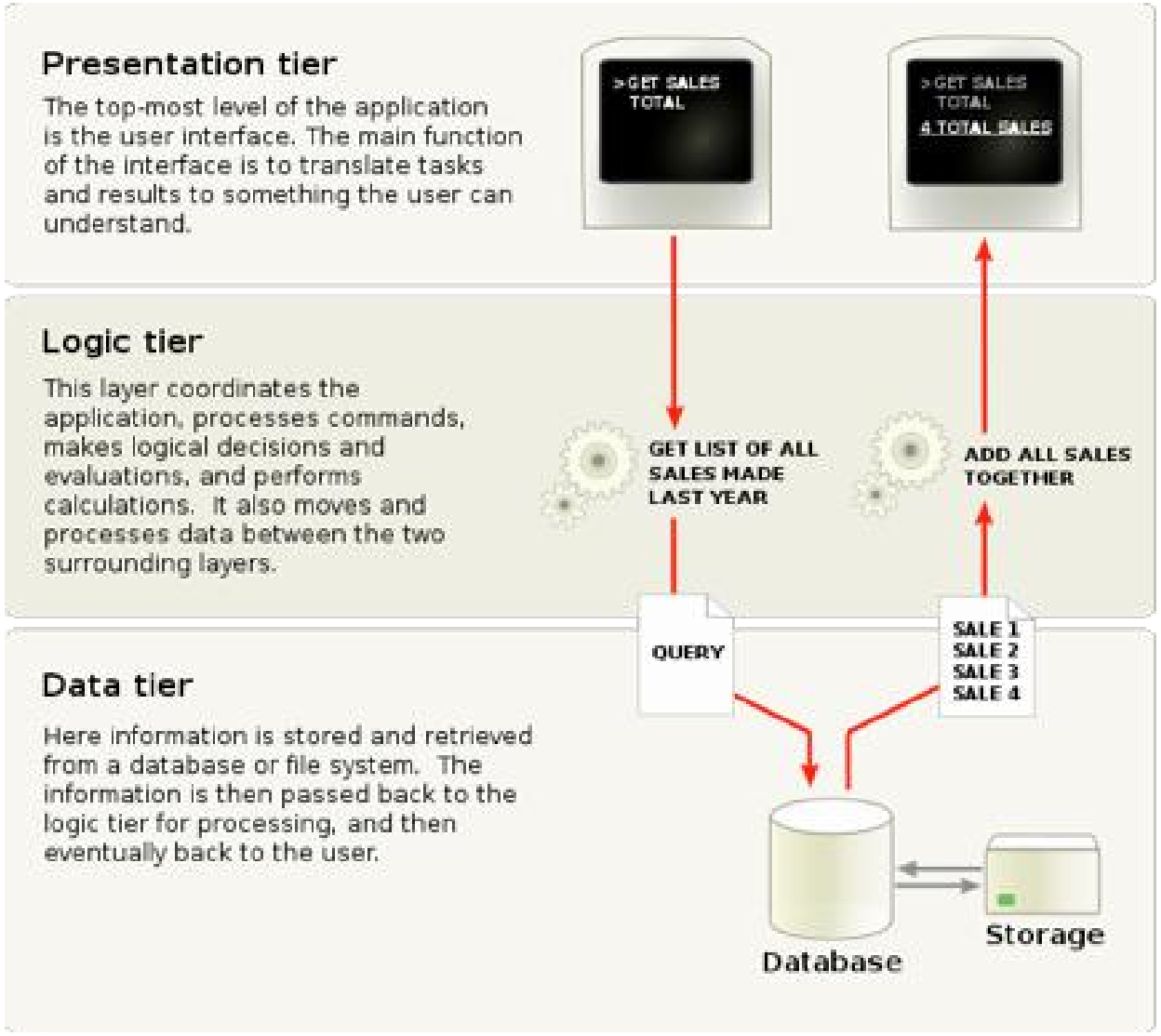
Why put images aside?

- Think about the scenarios:
  - Logo changes
  - How to easily update logos for millions of phons?



Design Philosophy

Mobile Application Layered Design



The Three-Tiered Architecture

- The concepts of the three-tiered architecture apply to many design scenarios
  - Keep the presentation separate so it's lightweight, easier to maintain, and can be tested separately
  - Keep the logic separate so you can change the logic as needed without having to change the presentation too much
  - Keep the data separate because you should NEVER build a system based on the current data values. Why?

Client/Server Architecture

Client/Server Architecture

- Mobile apps locally are great, but so much more can be done when apps are working with cloud services!
- The question is: how much processing / data should be done locally vs. in the cloud?

- Thick Client
  - Business and some data services on the phone itself
  - Good for apps that have to run “off the grid”
  - Examples?

- Thin Client
  - Most business and all data services on the server
  - Good for apps that require phone services, but does require Internet connectivity.
  - Examples?

- Which is better? Depends on the app and how it’s used!
  - computation vs communication. energy, time, bandwidth (Youtube)
  - privacy vs accuracy (Nest smart thermostat)

Pros and Cons of using server

- Pros
  - computation power (e.g. movies)
  - sharing (e.g., facebook, amazon)

- Cons
  - communication cost (e.g. battery, bandwidth)
  - privacy (e.g. waze)

Challenges in Mobile Development

Overview

- Limitations of the Wireless Network
  - heterogeneity of fragmented networks
  - frequent disconnections
  - limited communication bandwidth
- Limitations of the Mobile Devices
- Limitations of Battery

Frequent Disconnections

- Handoff blank out (>1*ms* for most cellulars)
- Drained battery disconnection
- Voluntary disconnection (turned off to preserve battery power, also off overnight)
- Roam-off disconnections

Limited Wireless Bandwidth

- Orders of magnitude slower than fixed network
- Higher transmission bit error rates (BER)
- Mutual interference
  - Difficult to ensure Quality of Service (QoS)
- Limited communication bandwidth exacerbates the limitation of battery lifetime.

Limitations of Mobile Devices

- Short battery lifetime and theft or destruction => unreliable
- Sometimes unavailable (disconnection or turned-off)
- Limited capability (display, computation, memory, input devices, and disk space)
- Device heterogeneity: phone, smartwatch, laptops, and other devices

Battery limitations

- Most resources increasing exponentially except battery energy

- Strategies:
  - Energy harvesting: Energy from vibrations, moving humans
  - Scale content: Reduce image, video resolutions to save energy
  - Auto-dimming: Dim screen whenever user not using it. E.g. talking on phone

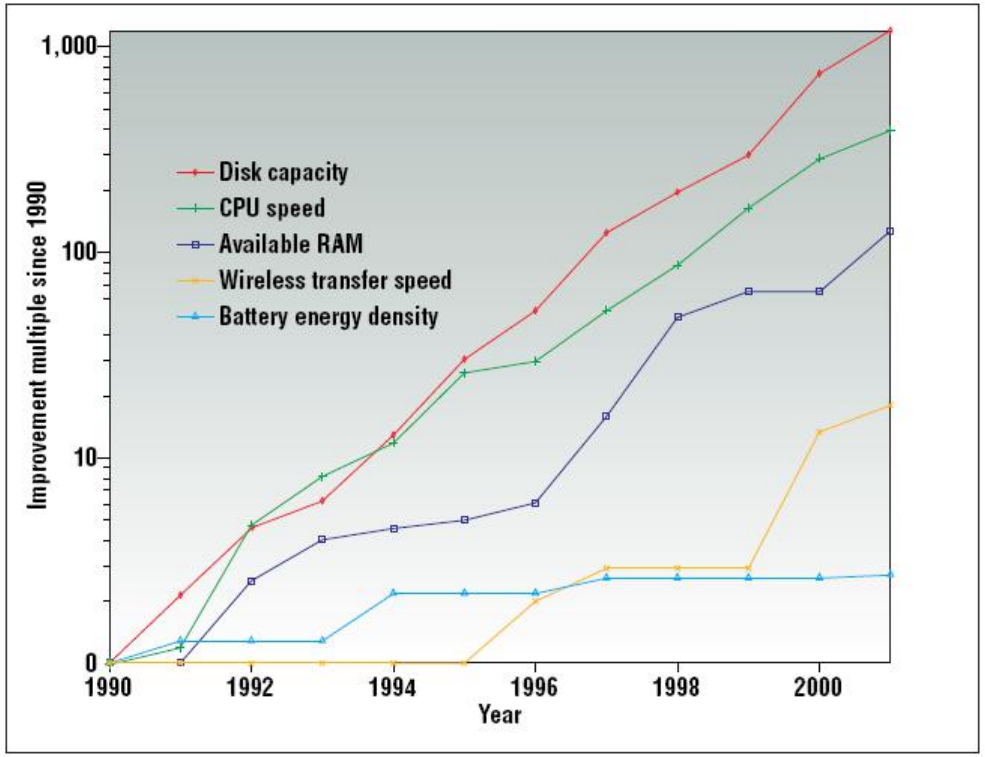


Figure 1. Improvements in laptop technology from 1990–2001.