APP PUBLISHING

There are several ways to publish an app:

- Release it for a predefined number of users (ie, apk download and communication per mail), often called sideloading.
- Release it on a marketplace like Google Play.

APP PUBLISHING: SIDELOADING

- Create a package file from your sources (Build -> Generate Signed Bundle / APK)
- Send the apk file per mail or put it on a web server.
- Inform the users that a new version is available.

APP PUBLISHING: GOOGLE PLAY STORE I

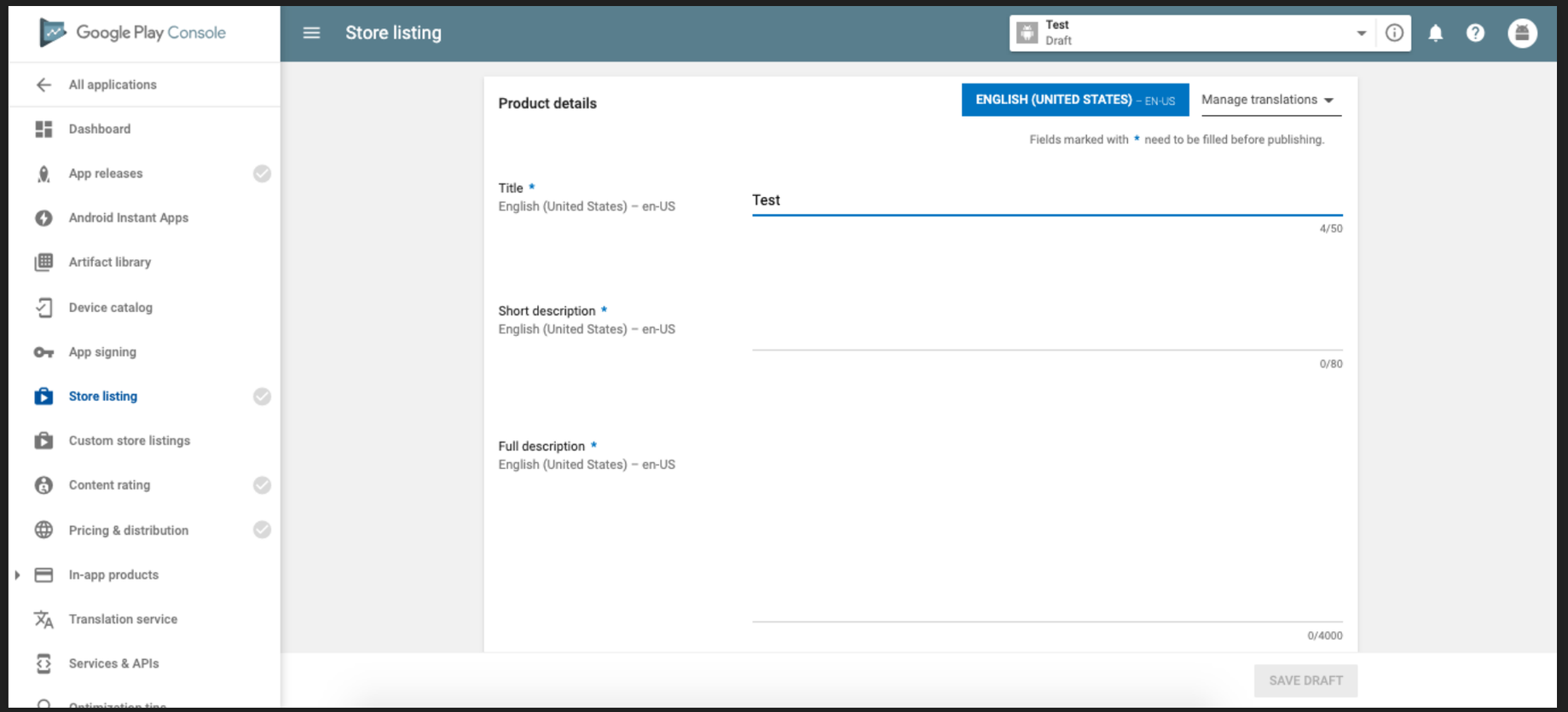
- Prepare your sources to be packaged.
- Version your app.

```
defaultConfig {
minSdkVersion 26
targetSdkVersion 30
versionCode 1 //int number (increase for new ver
versionName "1.0" //text to be displayed to the
...}
```

APP PUBLISHING: GOOGLE PLAY STORE II

- Sign your app.
- Prepare an app release in the Google Play Store console (provide a description, screenshots, icons, categories, pricing, etc.)
- Upload your app to Google Play Store

APP PUBLISHING: GOOGLE PLAY STORE CONSOLE



ANDROID INSTANT APP

- App that can be run without installing.
- Needs to have some special attributes in order to be an instant app.
- Min Api-Level: 21 (Android 5.0)

CREATE AN INSTANT APP

- Add:

```
<dist:module dist:instant="true" /> to
```

  to the app's manifest.

- Check the allowed permissions. You can use the camera or the internet but cannot create instant apps accessing local files.

- Test an instant app in Android Studio (enable "Deploy as instant app" in the General tab of Run/Debug).

INSTANT APP ADVANTAGES

- Can be tested before downloading
- Runs immediately
- Can be used even if the user doesn't have space on his/her device
- Can be shared easily

BLUETOOTH LE

WHAT IS BLUETOOTH LE?

- Bluetooth LE (sometimes called BLE) is a low power radio transmission standard.
- Range is roughly 10 meters.
- Since 4.0, it is part of the Bluetooth standard.
- Uses 2.4 GHz ISM band radio frequencies.
- The maximum amount of data is 1MB/s, the maximum transmit power is 10mW.

RADIO FREQUENCIES

- Radio Frequencies (short RF) are electromagnetic waves in the range from 3 kHz to 300 GHz.
- RFs are measured in Hz (1 Hz is 1 cycle per second).
- RFs are sent by a sender and received by a receiver who needs an antenna. Sender and receiver must send on the same frequency.
- Other important parameters when sending/receiving electromagnetic waves are the transmitter power, the size of the antenna, noise and interference signals.

FREQUENCIES AND DATA

Some common rules:

- The higher the frequency the more data you can transmit.
- The larger the frequency band the more data you can transmit.
- The lower the frequency the larger the range (by constant transmitter power).

COMMON FREQUENCIES I

RF	Service
50Hz	AC power grid
500KHz	International SOS Signal
75KHz	Radio Clock Broadcasting Signal HBG (shut down)
77,5KHz	Radio Clock Broadcasting Signal DCF77
300-3000KHz	AM Radio

COMMON FREQUENCIES II

RF	Service
87,5 MHz - 108,0 MHz	FM Radio
174-240 MHz	DAB Radio
900 MHz	First GSM Networks
2,455 GHz	Microwave oven

FREQUENCIES IN 2.4 GHZ

- Can be used from 2400-2500 MHz, however, in most countries, the band from 2483.5 MHz – 2500 MHz has strict emission limits.
- This band is jammed by microwave ovens, this was the main reason why it was not used for cellular data transmission.
- Are used by a number of different services (see next slide), therefore, collision protection is important.

SERVICES IN 2.4 GHZ

- DECT (cordless phone)
- Bluetooth
- Wifi (newer Wifi standards might use a different band like 5 GHz)
- Wireless USB
- ZigBee

BLUETOOTH CHANNELS

- BLE uses 2401 - 2481 MHz.
- Most services divide the band into channels. Some services use overlapping channels, others (like Bluetooth LE) nonoverlapping channels.
- Bluetooth divides the band into 79 channels (each 1 MHz wide) and changes channels up to 1600 times per second.
- BLE uses 40 channels with 2 MHz each.
- There are three advertising channels, the rest (37) are data channels.

BLUETOOTH FREQUENCIES I

Channel	Lower Frequency	Upper Frequency
0	2403	2405
1	2405	2407
…		
36	2477	2479
37*	2401	2403
38*	2425	2427
39*	2479	2481

WIFI FREQUENCIES I (FOR COMPARISON)

Channel	Lower Frequency	Upper Frequency
1	2401	2423
2	2404	2428
...		
12	2456	2478
13	2461	2483
14	2473	2495

WIFI FREQUENCIES II

Country/Continent	Permitted Channels
Japan	1 to 14
Europe	1 to 13
Asia (excl. Japan/Taiwan)	1 to 13
North America	1 to 11 *
South America	1 to 11 *
Taiwan	1 to 11 *

* Some countries allow channels 12 and 13 in low power mode.

BLUETOOTH PROFILES

- Standard Bluetooth provides different (application specific) profiles that specify how the data is transmitted between two devices and what functions exist.
- Bluetooth LE only has one profile, the Generic Attribute Profile (GATT). GATT itself contains numerous sub profiles.
- Sub profiles include: Health Thermometer Profile, Heart Rate Profile, Phone Alert Status Profile and the Proximity Profile (PXP, used by Beacons).

PXP PROFILE

- PXP enables proximity monitoring between two devices.
- Contains two roles: The proximity reporter (the server) and the proximity monitor (the client).

http://stackoverflow.com/questions/20352200/android-ble-retrieve-service-uuid-in-onlescan-callback-when-advertised-from

BEACONS

WHAT IS A BEACON?

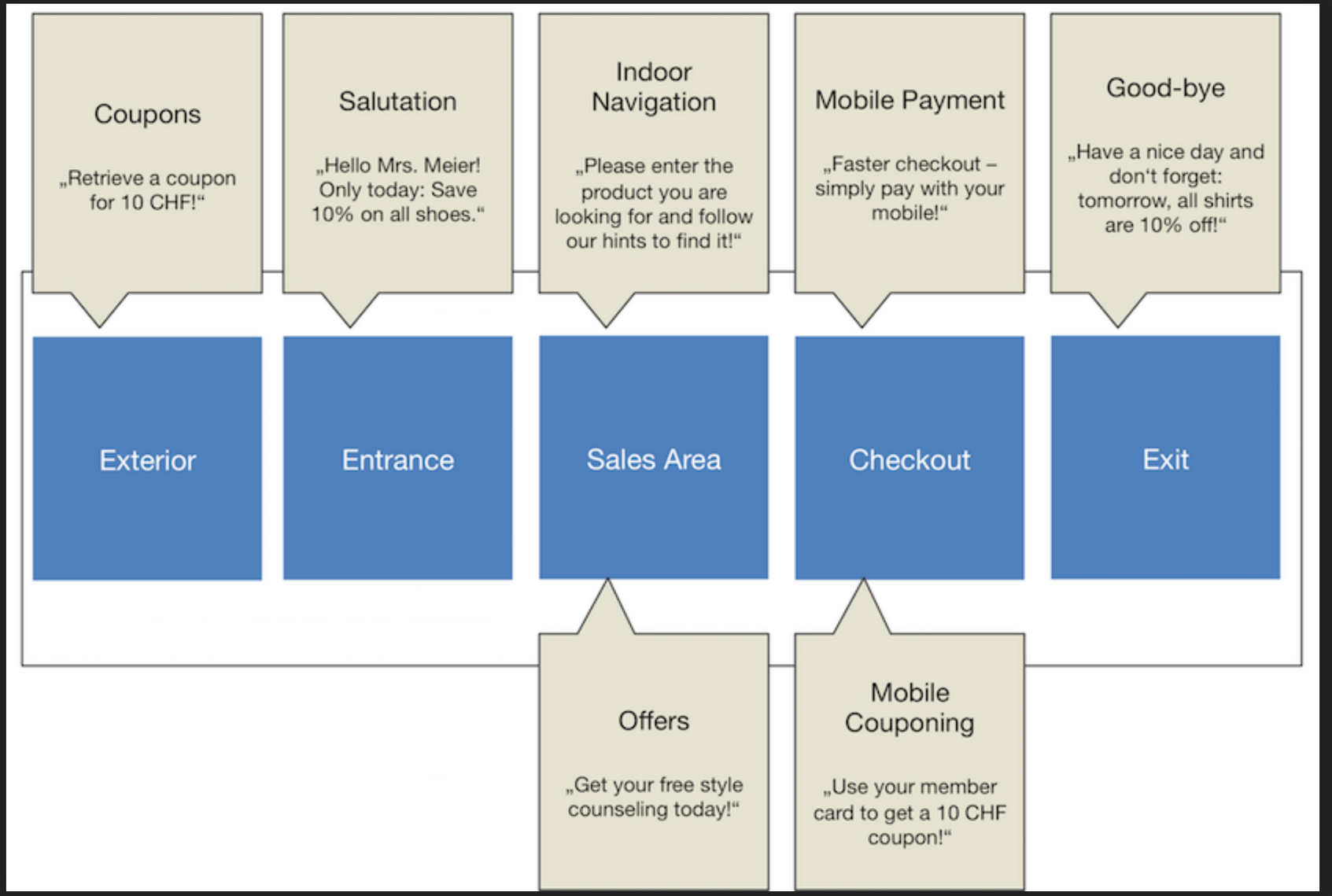
- Small, low powered device that transmits small amounts of data via Bluetooth Low Energy (BLE) up to 50 meters
- Typically powered by a battery
- Typically used to enhance services that trigger certain actions whenever the user is in range of a beacon

BEACON STANDARDS

- There are three different beacon standards: iBeacon, Eddystone and Altbeacon. In practice, iBeacon is most popular and will be covered within this presentation.
- iBeacons are ordered in regions. Each region has a unique ID.
- Apps can be informed when entering / exiting a region.
- Within a region, apps can scan for all beacons of the region and retrieve their minor/major numbers as well as their received signal strength indicator (RSSI).

BEACONS AND PRIVACY

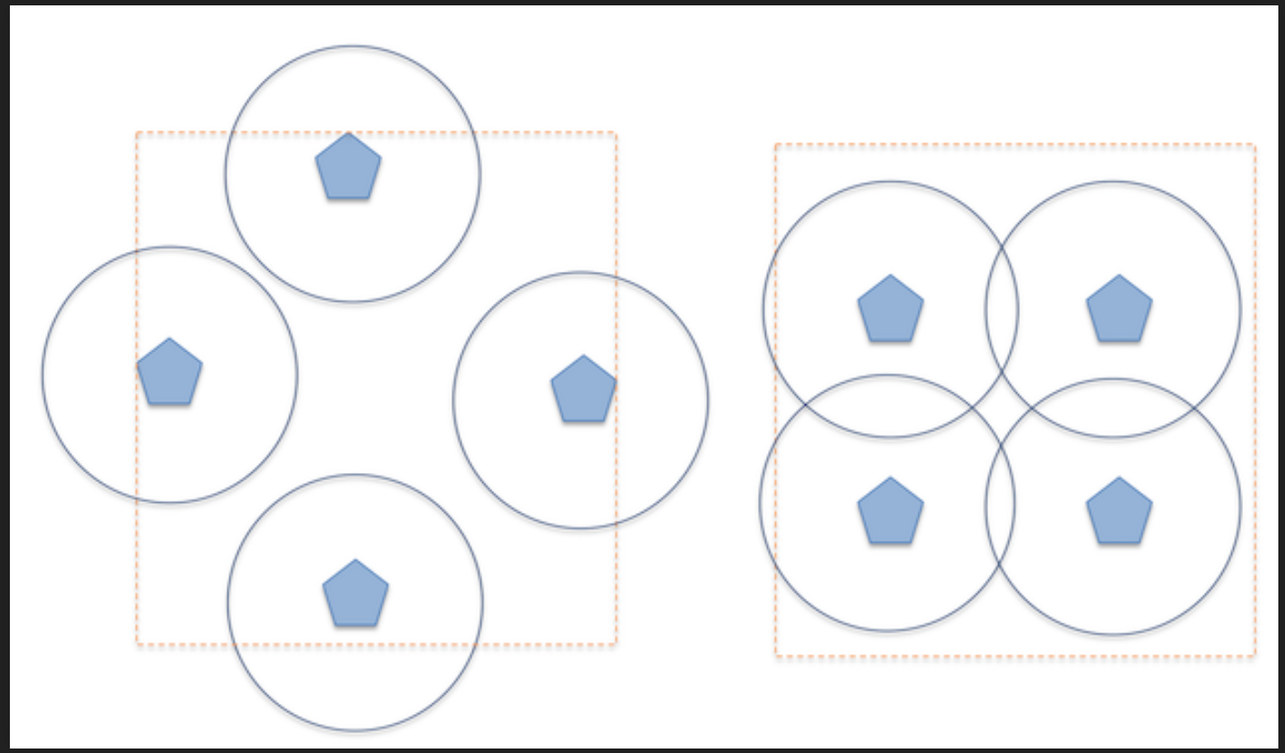
- Privacy concerns are often mentioned when it comes to beacons. However, the beacon itself is only a sender that sends a number frequently.
- In combination with an app, user tracking is possible as the app could send this number, together with a timestamp, to a central server.
- Users can disable beacon tracking in Android and iOS.



OTHER SCENARIOS

- Museum: Antwerp Museum
- Travel: How iBeacons will change the Travel Industry
- Events: 9 Ways iBeacon Will Transform Events

INTERIOR / EXTERIOR



Typical Placement outside / inside

APPLE AIRTAGS

- Can be used to tag real world items (keys, handbags,...). It is possible to show the last position of an item.
- A user will be warned whenever the tag is not in range anymore.
- Tags are received by other Apple devices and automatically transferred to the apple server. This wil update the position of a lost item.

AirTags at Apple

Sending AirTags around the world

TAG SENSORS

A number of companies offer improved tag that can be used to measure a number of additional parameters:

- Temperature
- Humidity
- Air Pressure
- Rotation
- Pressure
- ...

Examples are Ruuvi and PuckJS.

SOURCE IN ANDROID I

Use a library like altbeacon:

```
implementation 'org.altbeacon:android-beacon-library:2+'
implementation
  'androidx.localbroadcastmanager:localbroadcastmanager:1.0.0'
```

Implement the receiving of beacon data in an Activity:

```
class MainActivity : AppCompatActivity(), BeaconConsumer {
  private var beaconManager: BeaconManager? = null
  ...
```

SOURCE IN ANDROID II

Location and Internet permissions are needed:

```
<uses-permission android:name=
  "android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
```

```
if (this.checkSelfPermission(
  Manifest.permission.ACCESS_FINE_LOCATION)
  == PackageManager.PERMISSION_DENIED) {
  ActivityCompat.requestPermissions(this, arrayOf(
  Manifest.permission.ACCESS_FINE_LOCATION), 1)
}
else {initBeaconManager()}
```

SOURCE IN ANDROID III

```
beaconManager = BeaconManager.getInstanceForApplication(this)

//this call is needed to find iBeacons (by default, only Altbeacons
//will be found)
beaconManager!!.beaconParsers.addBeaconParser()
  .setBeaconLayout("m:2-3=0215,i:4-19,i:20-21,i:22-23,p:24-24"))

//now bind the beaconManager to the activity.
beaconManager?.bind(this);
```

SOURCE IN ANDROID IV

```
//this method will be called when the binding is done
override fun onBeaconServiceConnect() {
 //init the notifier
 beaconManager!!.addRangeNotifier { beacons, region ->
    Log.i(TAG, "Range notifier called ${beacons.size}
       ${region.uniqueId}")
    for (beacon in beacons) {
        Log.i(TAG, "Beacon found:
           distance (m)=${beacon.distance}
           major=${beacon.id2} minor=${beacon.id3}")
    }
 }
```

SOURCE IN ANDROID IV

```
//and start the scan
 try {
    Log.i(TAG, "start ranging beacons in region")
    beaconManager!!.startRangingBeaconsInRegion(
    //this is the UUID of the beacon region
    Region("F7826DA6-4FA2-4E98-8024-BC5B71E0893E", null, null, null))
 } catch (e: RemoteException) {
    Log.e(TAG, e.toString())
 }
}
```

SOURCE IN IOS I

```
//define a manager and region
var manager : CLLocationManager?
var region : CLBeaconRegion?

//and create them
manager = CLLocationManager()

manager?.requestAlwaysAuthorization()

//define the UUID of the region to scan
let proximityUUID : NSUUID = NSUUID(UUIDString:"F78...")!
```

SOURCE IN IOS II

```
//create an instance of the region with the UUID
region = CLBeaconRegion(proximityUUID:proximityUUID,
    identifier:"My Region")

manager?.delegate = self

//start both scans (region monitoring and beacon ranging)
//you can split these commands and, for example, ask for
//region monitoring and start the beacon ranging when a region
//has been found
manager?.startMonitoringForRegion(region!)
manager?.startRangingBeaconsInRegion(region!)
```

SOURCE IN IOS III

```
//implement the methods from the delegate
func locationManager(manager: CLLocationManager,
    didEnterRegion region: CLRegion) {
    log("didEnterRegion")
}


func locationManager(manager: CLLocationManager,
    didExitRegion region: CLRegion) {
    log("didExitRegion")
}
```

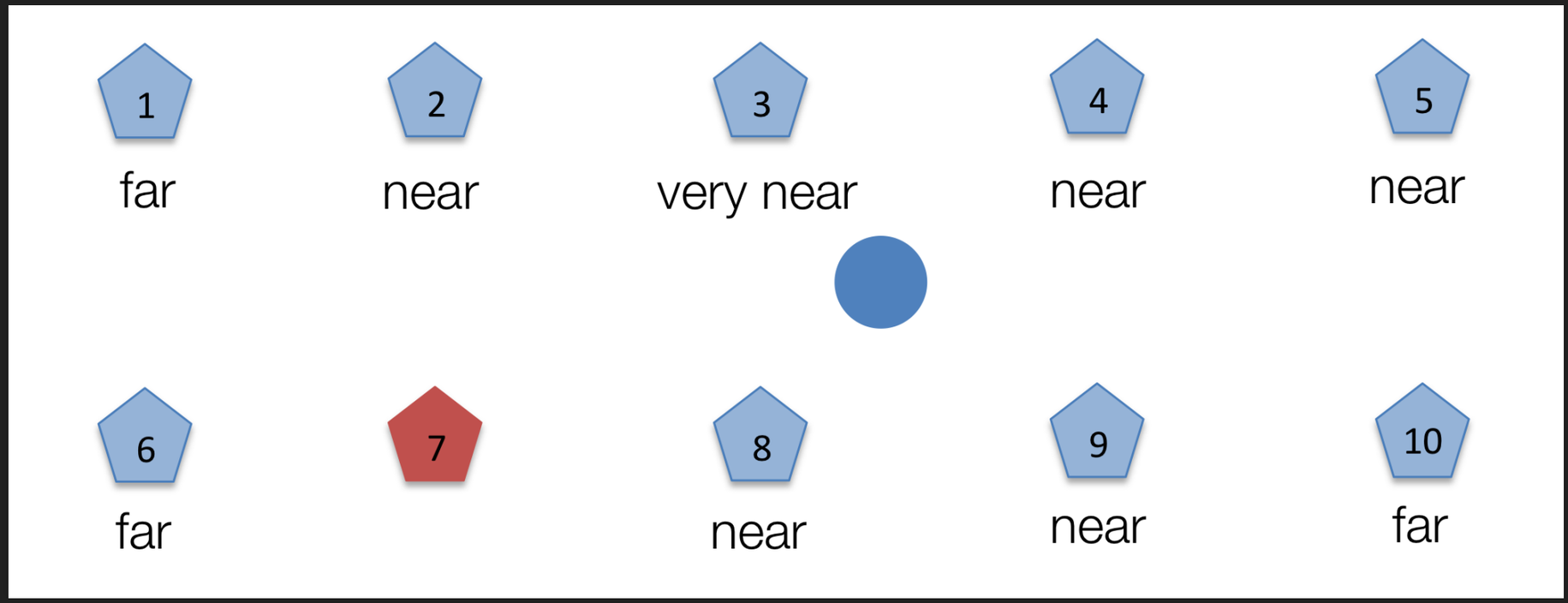
SOURCE IN IOS IV

```
//this method is called when the ranging of beacons has been
//done.
func locationManager(manager: CLLocationManager,
    didRangeBeacons beacons: [CLBeacon],
    inRegion region: CLBeaconRegion) {
    log("didRangeBeacons");
    for beacon in beacons {
      log("Beacon " + String(beacon.minor) +
        String(beacon.major) + String(beacon.rssi));
    }
}
```

BEACON ERROR DETECTION

Problem: The battery for typical beacons only last one to two years.

Fortunately, there is an easy way to find beacons with an empty battery in most scenarios.



Write an app that recognizes if you are in the front or back of the classroom by scanning for two beacons.

APPS FOR CARS

APPS FOR CARS I

- Both Google and Apple provide solutions for integrating a mobile device in a car.
- Both solutions work quite similar: Connect your phone with your car and use a number of features on a special display.
- Apples solution is CarPlay, Google has developed Android Auto.

APPS FOR CARS II

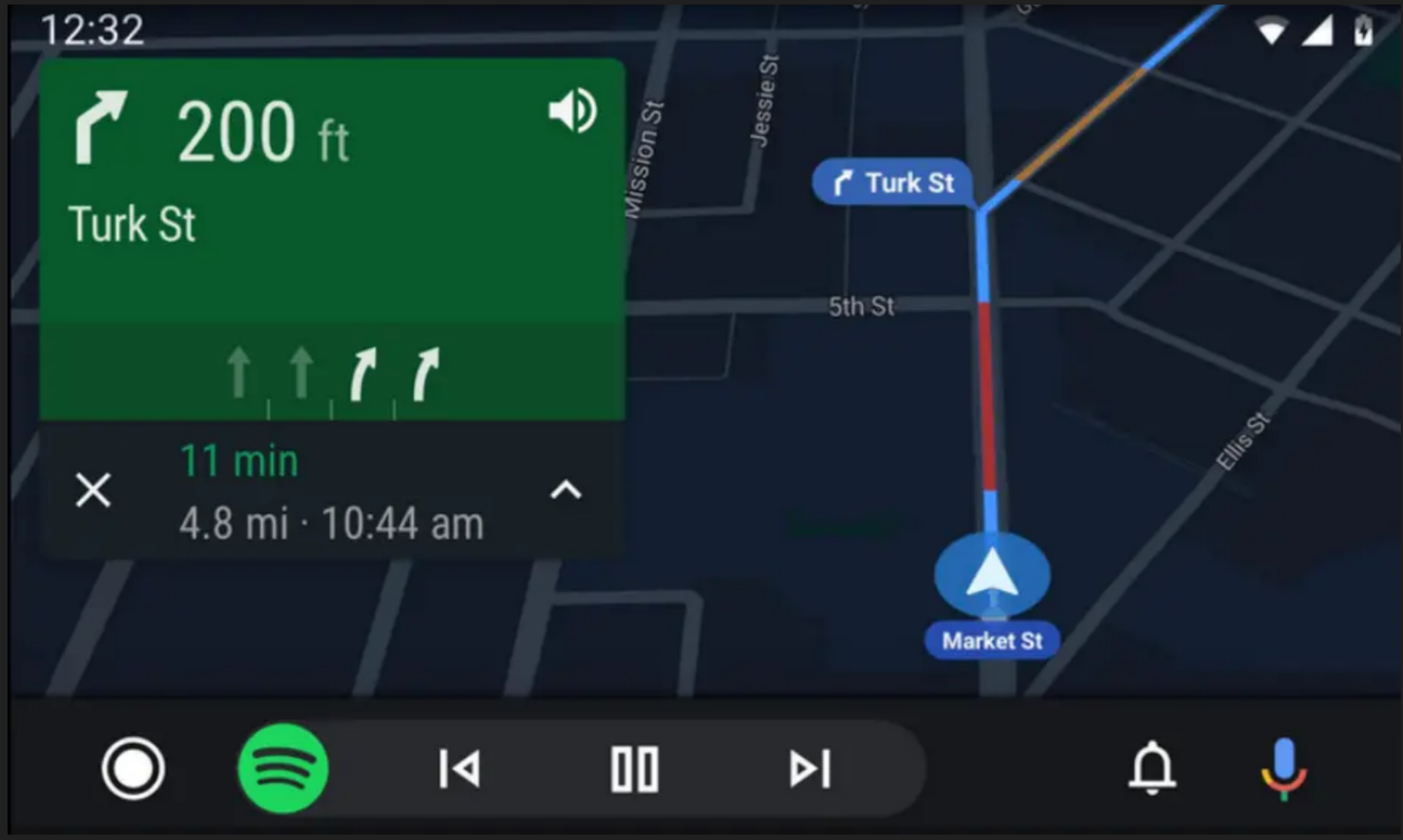
- All apps are running on the phone but use the car's dash as display.
- You can interact with an app by using the vehicles controls (touch screen, buttons, etc).

START ANDROID AUTO SIMULATOR

- Install the Android Auto companion app from the Play Store and run it.
- Install the Android Auto Desktop Head Unit Emulator from Android Studio SDK Tools.
- Make a tcp 5277 port forward.
- Run the desktop head unit (DHU) on your stationary machine.

Details can be found here.

ANDROID AUTO DISPLAY



CARPLAY

APPLE CAR KEY I

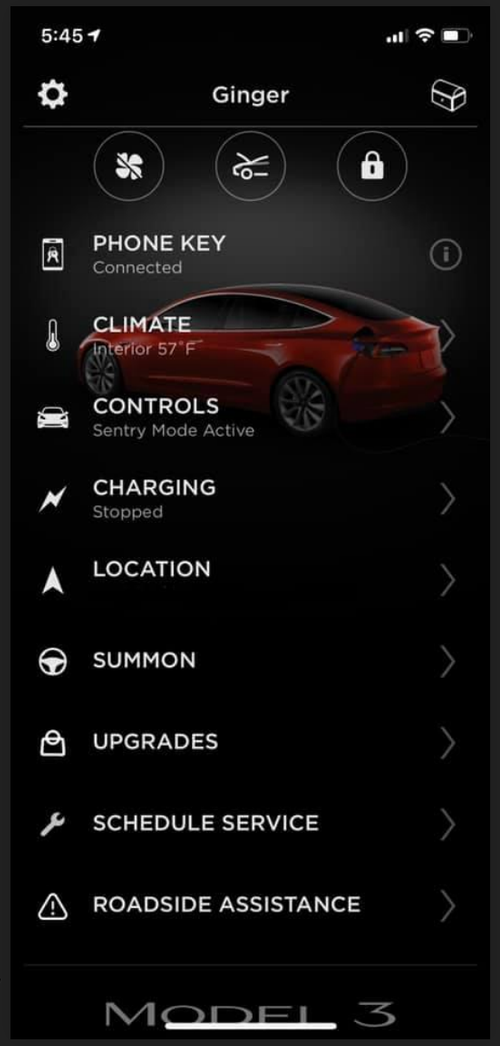
- Beginning with iOS14, the iPhone (and Apple Watch) can be used as a car key. This is the first feature that extends the Apple CarPlay approach to a wider interface with the car (until then, CarPlay could be seen as a big display without any car communications).
- After registering the car to the Apple account, the key itself is is stored in Apple Wallet. It is possible to share the key with other persons.
- Communications between the car and the iOS device is done with NFC.

APPLE CAR KEY II

TESLA APP

As Tesla offers their own app to access internal functions of the car the Tesla app shows a number of interesting features:

- control almost everything from your phone, from door locks to the engine to the doors.
- get the values of a number of sensors, from battery to location.
- access service documents and communicate with a service station in case of an emergency.



CREATING AN ANDROID AUTO MEDIA PLAYER

- Override MediaBrowserService
- Register the service in your manifest
- Implement methods to display a menu tree as well as play a media file

AUTO MEDIA PLAYER I

```
<service android:name=".MyMediaBrowserService"
  android:exported="true">
  <intent-filter>
   <action android:name="android.media.browse.MediaBrowserService"/>
  </intent-filter>
</service>
<meta-data android:name="com.google.android.gms.car.application"
  android:resource="@xml/automotive_app_desc"/>
```

```
<automotiveApp> <!-- register app as media service ())
    <uses name="media"/>
</automotiveApp>
```

AUTO MEDIA PLAYER II

```
class MyMediaBrowserService : MediaBrowserService() {
  private var mMediaSession: MediaSession? = null
  override fun onCreate() {
    mMediaSession = MediaSession(this, "Android Auto Audio Demo")
    mMediaSession!!.isActive = true
    mMediaSession!!.setCallback(mMediaSessionCallback)
```

AUTO MEDIA PLAYER III

```
override fun onGetRoot(
  clientPackageName: String, clientUid: Int,
  rootHints: Bundle?
): BrowserRoot? {

  //return our root element (never displayed)
   return BrowserRoot(MY_MEDIA_ROOT_ID, null)
}
```

AUTO MEDIA PLAYER IV

```
override fun onLoadChildren(
  parentMediaId: String,
  result: Result<List<MediaBrowser.MediaItem>>) {
    val mediaItems: MutableList<MediaBrowser.MediaItem> =
            ArrayList()

  // Check if this is the root menu
  if (MY_MEDIA_ROOT_ID == parentMediaId) {
      for (mp3Entry in media) {
            mediaItems.addAll(createMediaData())}
  } else {
      // generate submenu...
  }
  result.sendResult(mediaItems)
}
```

AUTO MEDIA PLAYER V

```
fun createMediaData() : MutableList<MediaBrowser.MediaItem> {
  val mediaId = mp3Entry.key
  val mediaItem = mp3Entry.value
  val mediaDescription = MediaDescription.Builder()
      .setTitle(mediaItem.title)
      .setMediaId(mediaId)
      .build()
  mediaItems.add(
    MediaBrowser.MediaItem(
      mediaDescription,
      MediaBrowser.MediaItem.FLAG_PLAYABLE)}
```

Suppose you are a Google Manager and want to push Android Auto. What would be a good strategy?