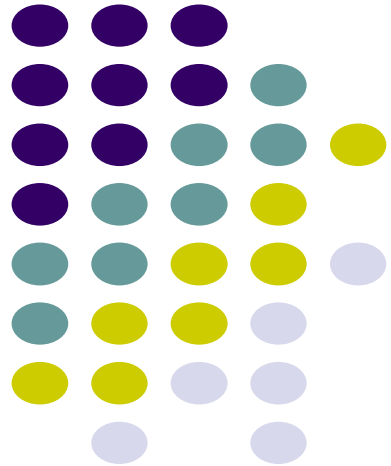
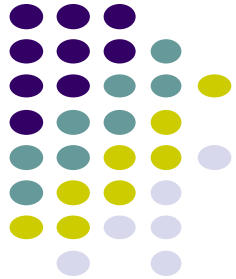
CS 528 Mobile and Ubiquitous Computing

Lecture 7a: Maps, Other Android Mobile Components and Activity Recognition

Emmanuel Agu





Using Maps

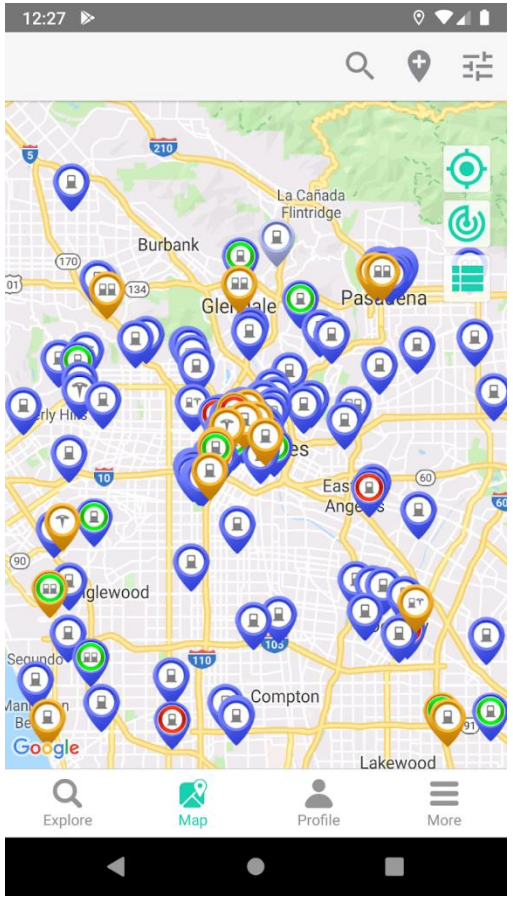
Announcements

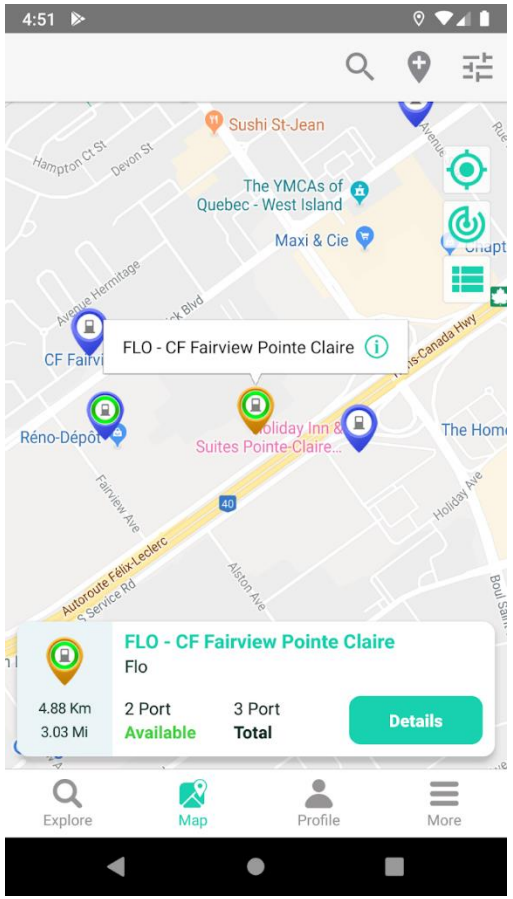
- No class next Thursday (October 19): Term break!

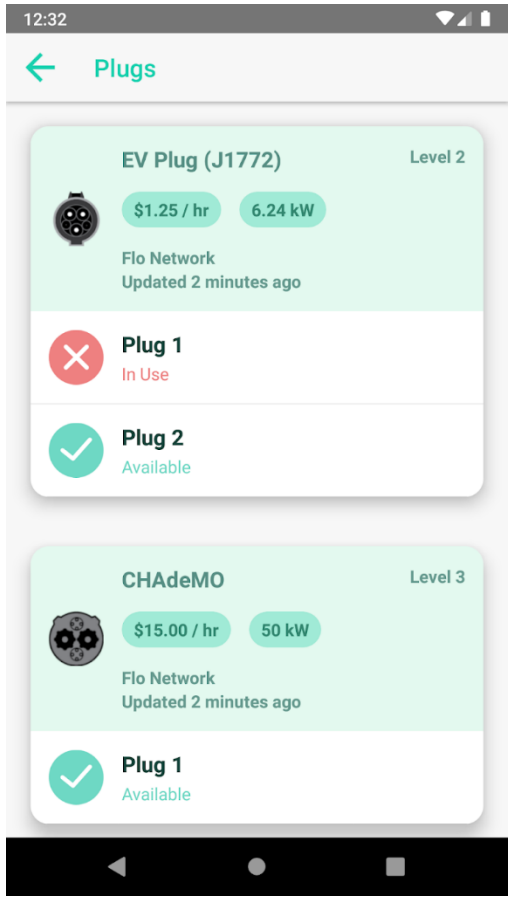
- Reminder: Teams will present research papers on Nov 16 and Nov 30
  - I have posted papers that teams can select to present
  - Papers will be assigned on first ask, first selected basis
  - Deadline to select papers to present: by class time on Thursday, November 9, 2023

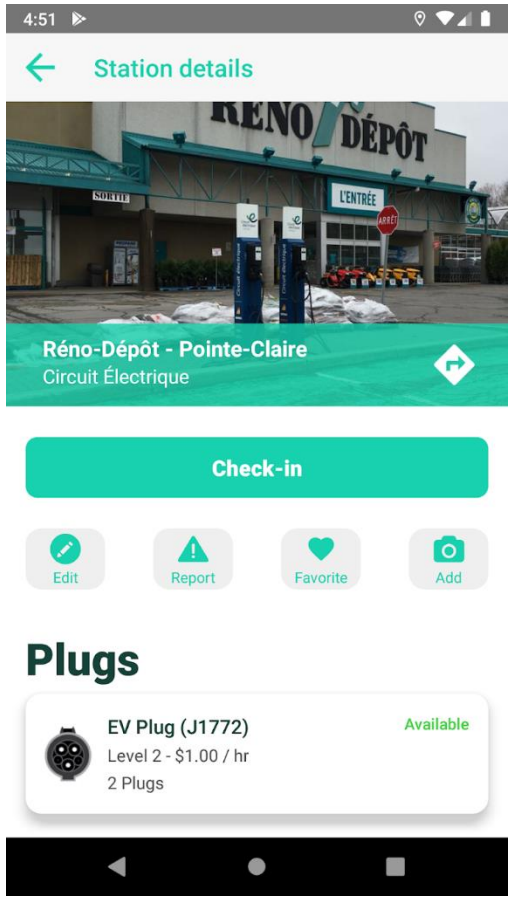
Example App that uses Maps: ChargeHub EV

- Displays location of all Electric Vehicle Charging stations on map





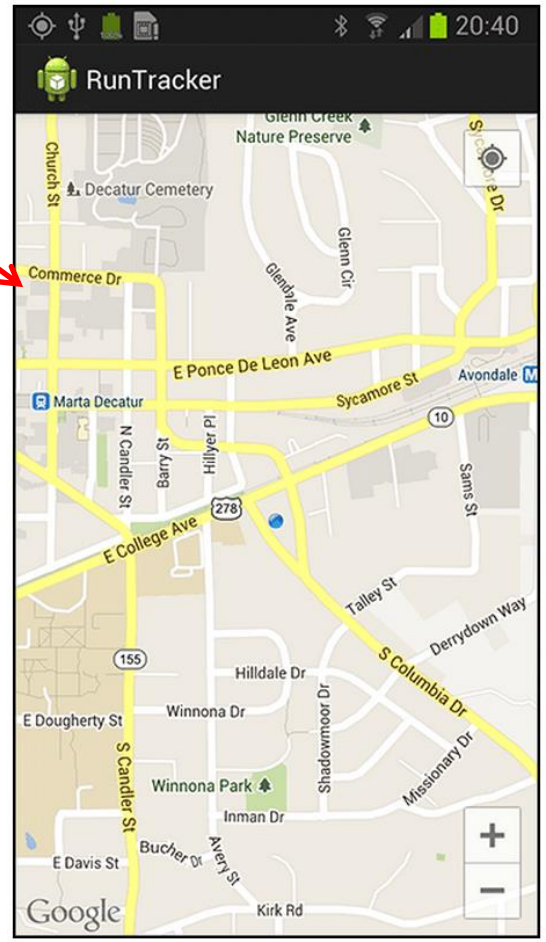


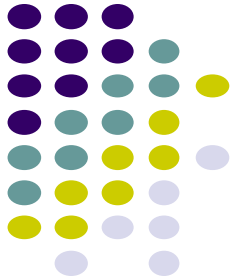


Maps SDK for Android QuickStart

https://developers.google.com/maps/documentation/android-sdk/start

- SDK has QuickStart basic maps template, easy setup
- 2 main files:
  - **Activity_maps.xml:** XML file
    - Contains UI widget that displays maps
  - **MapActivity.kt** kotlin file (extends Activity),
    - Handles map-related lifecycle and management for displaying maps.





Steps for using Google Maps Android API

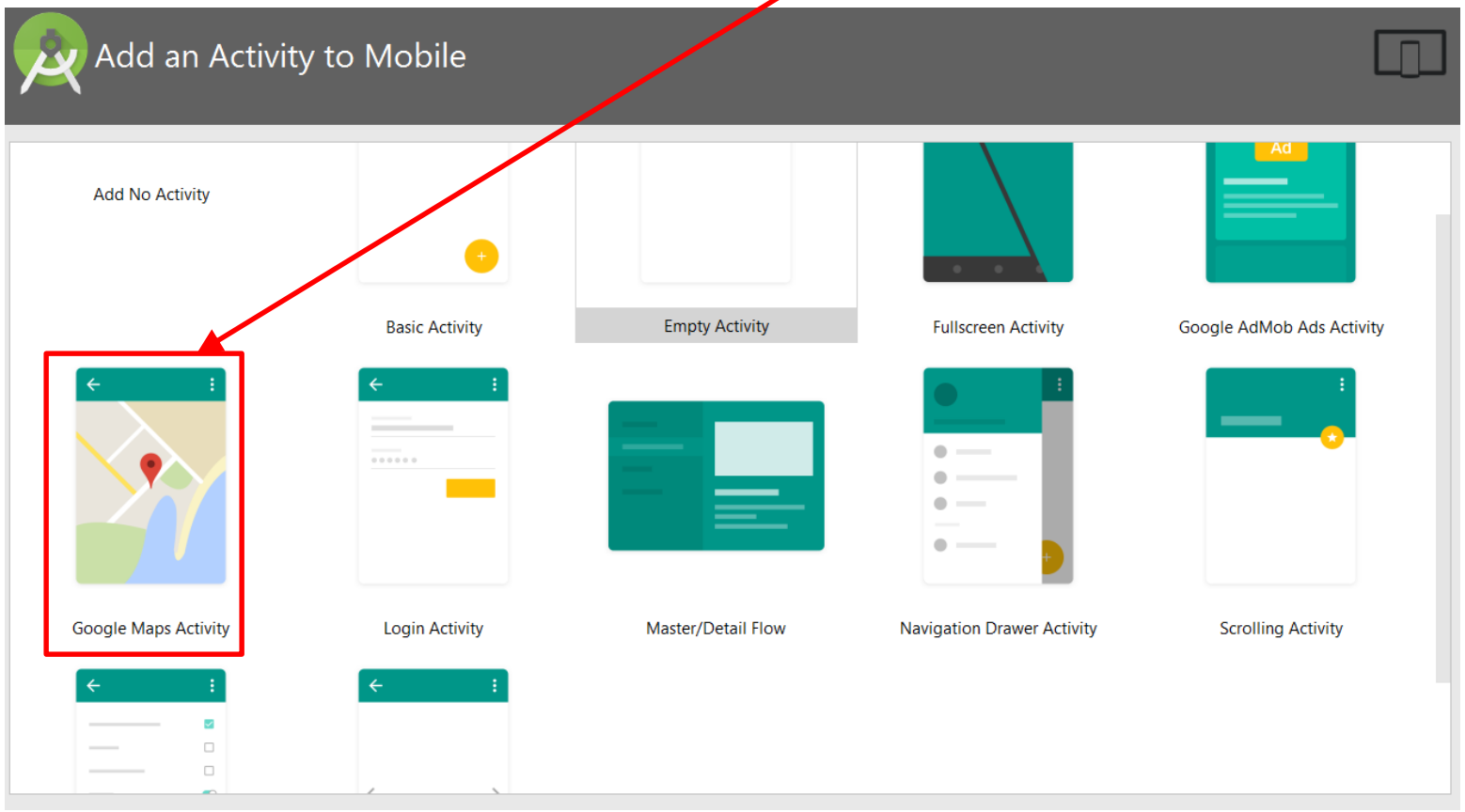
https://developers.google.com/maps/documentation/android-sdk/start

1. Set up development environment (Done!!)
   - Install Android studio https://developer.android.com/studio,
   - Update Gradle
2. Set up Android Device or Android emulator (Android 4.0 or later)
3. Create a Google Maps project in Android Studio
4. Set up Google Cloud project
5. Obtain Google Maps API key, add it to app
6. Hello Map! Take a look at the code
7. Connect an Android device
8. Build and run your app

Step 3: Create new Android Studio Project

https://developers.google.com/maps/documentation/android-api/start

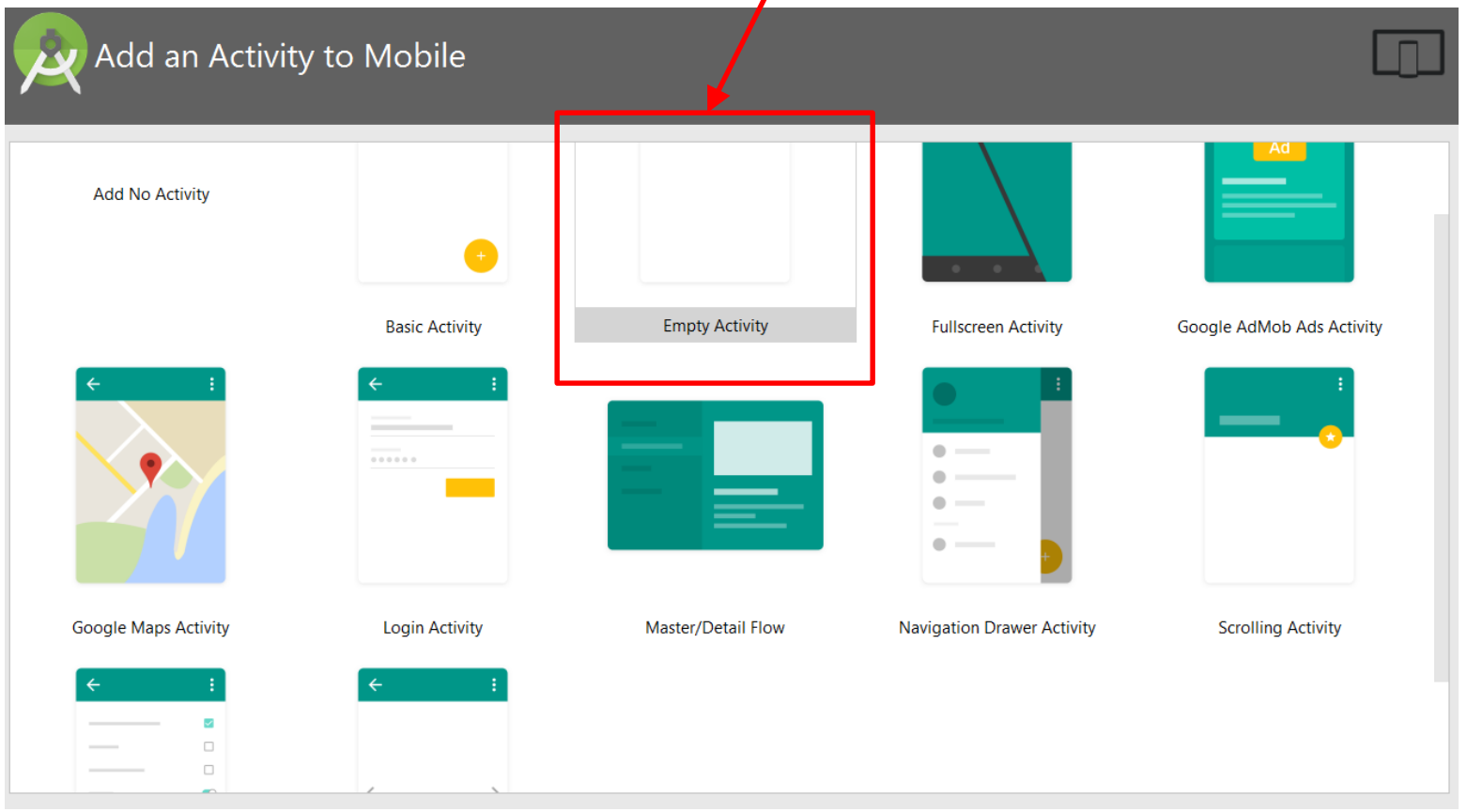
- On Android Studio Electric Eel or earlier, select “Google Maps Activity”, click Finish



Step 3: Create new Android Studio Project

https://developers.google.com/maps/documentation/android-api/start

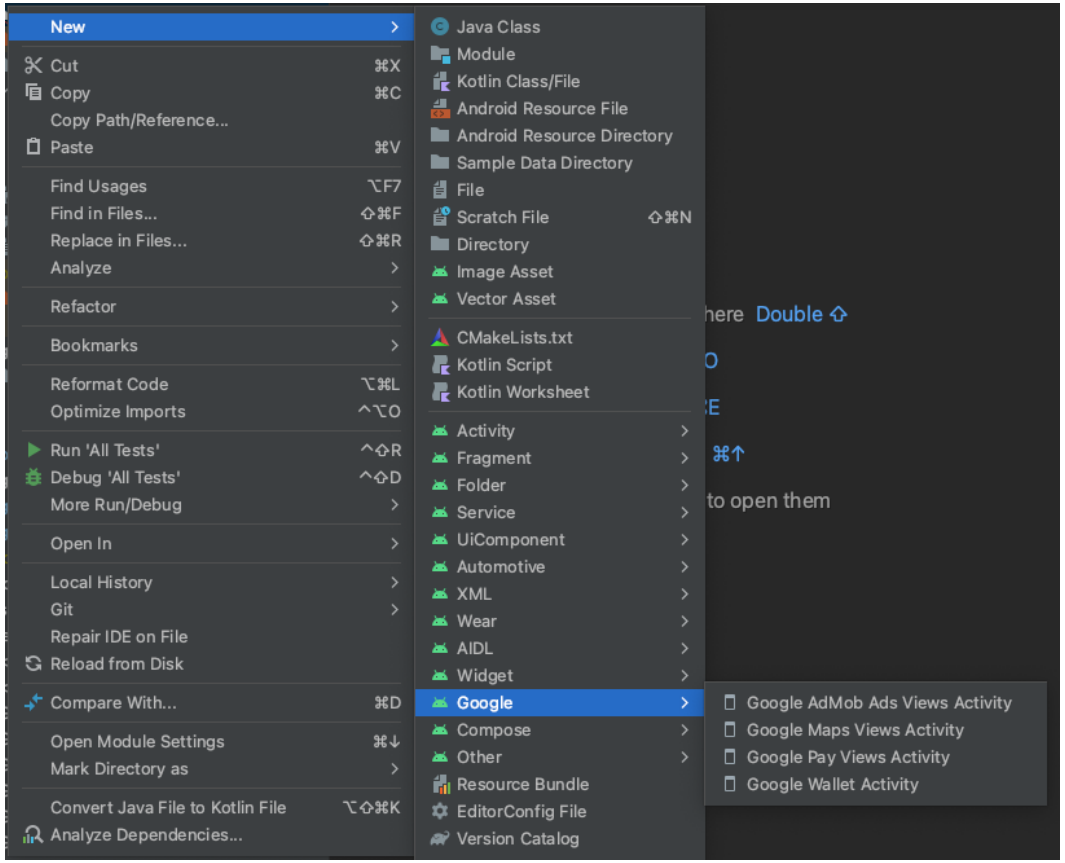
- On most recent Android Studio, select “Empty Activity”, click Finish



Step 3: Create new Android Studio Project

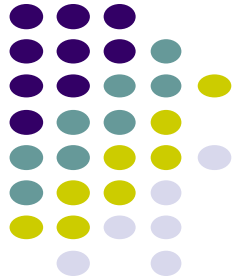
https://developers.google.com/maps/documentation/android-api/start

- On most recent Android Studio version
- Complete Google Maps Activity form:
  - Set language to kotlin
  - Set minimum SDK
- Click finish!
- Android Studio opens AndroidManifest.xml and MapsActivity files
- Add Google Maps Views Activity
  - Right-click on appropriate package
  - Select New -> Google -> Google Maps Views Activity



Background: API Key

https://developers.google.com/maps/documentation/android-api/start



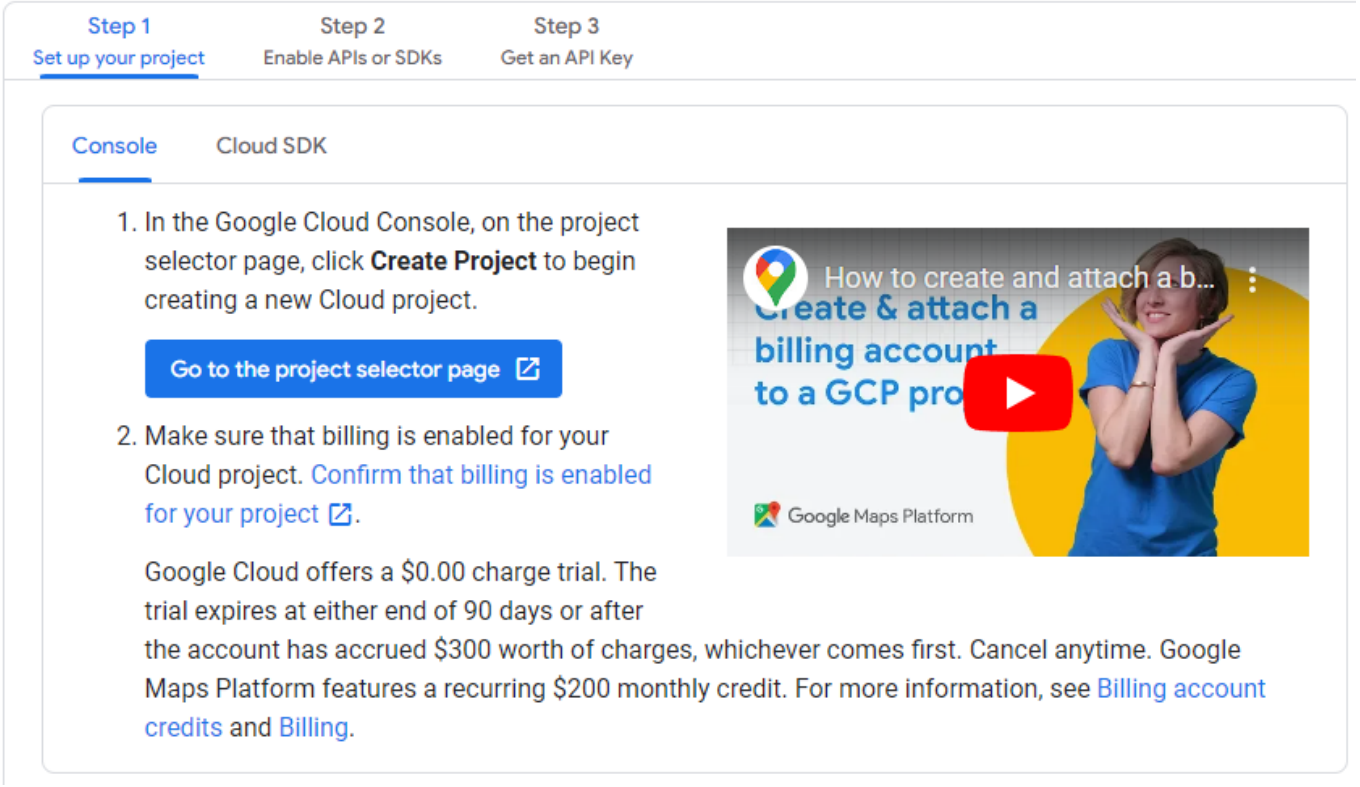
- To access Google Maps servers using Maps API, must add Maps API key to app
- Maps API key is free. E.g.

- Google uses API key to uniquely identify your app, track its resource usage, etc.
- Create Maps API key on Google cloud console

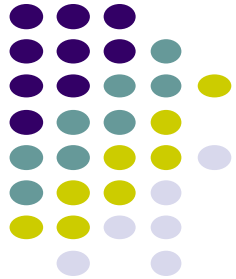
Step 4 & 5: Set up Google Cloud Project

https://developers.google.com/maps/documentation/android-sdk/start

- Complete cloud console setup steps (including getting API key)
- You need to log into google cloud console (https://console.cloud.google.com/) using your gmail account.
- Follow steps to add API KEY to your app



Step 6: Examine Code Generated by Android Studio Maps Template

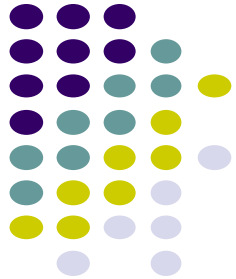


- activity_maps.xml file that defines layout is in **res/layout/activity_maps.xml**

Set default activity of fragment to MapsActivity

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/map"
    tools:context=".MapsActivity"
    android:name="com.google.android.gms.maps.SupportMapFragment" />
```

Step 6: Examine Code Generated buy Android Studio Maps Template



- Default Activity file is **MapActivity.kt**

```
internal class MapsActivity : AppCompatActivity(), OnMapReadyCallback {

    private lateinit var mMap: GoogleMap

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_maps)
        // Obtain the SupportMapFragment and get notified when the map is ready to be used.
        val mapFragment = supportFragmentManager
            .findFragmentById(R.id.map) as SupportMapFragment
        mapFragment.getMapAsync(this)
    }

    /**
     * Manipulates the map once available.
     * This callback is triggered when the map is ready to be used.
     * This is where we can add markers or lines, add listeners or move the camera. In this case,
     * we just add a marker near Sydney, Australia.
     * If Google Play services is not installed on the device, the user will be prompted to install
     * it inside the SupportMapFragment. This method will only be triggered once the user has
     * installed Google Play services and returned to the app.
     */
    override fun onMapReady(googleMap: GoogleMap) {
        mMap = googleMap

        // Add a marker in Sydney and move the camera
        val sydney = LatLng(-34.0, 151.0)
        mMap.addMarker(MarkerOptions()
            .position(sydney)
            .title("Marker in Sydney"))
        mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney))
    }
}
```

Steps 7, 8

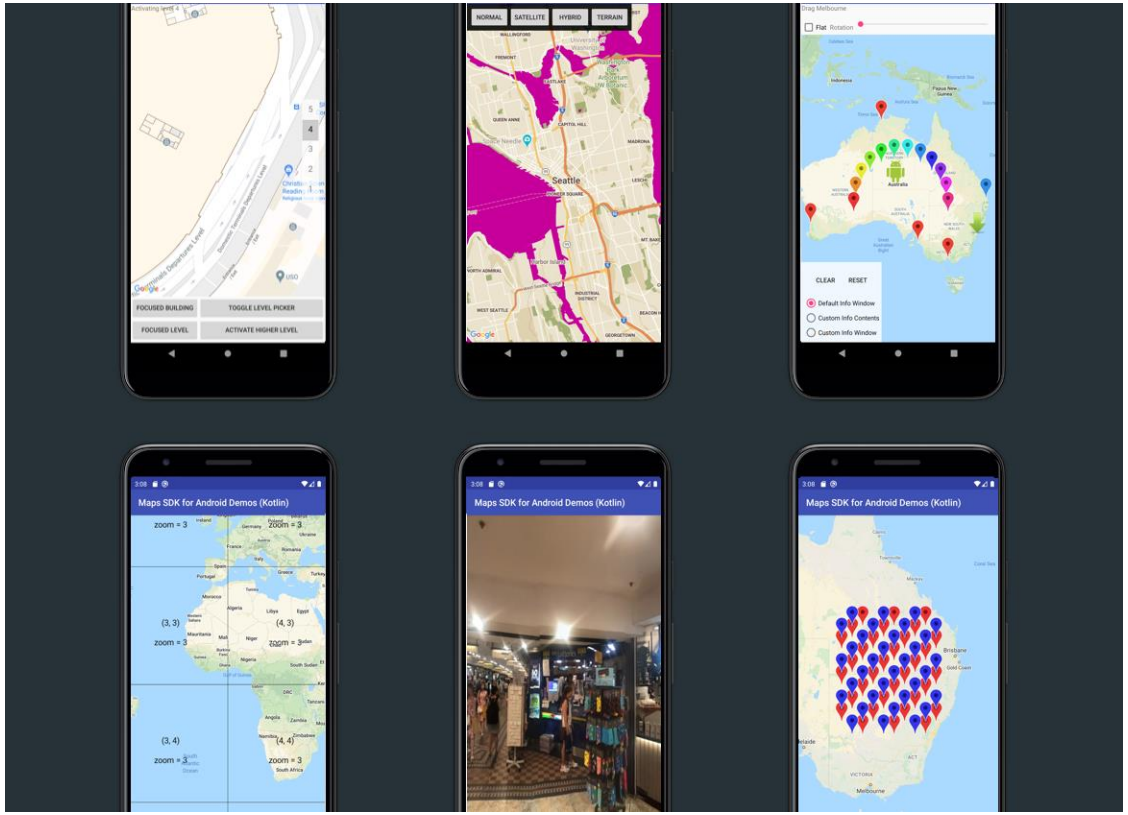
- **Step 7:** Connect to an Android device (smartphone)

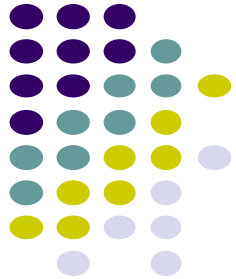
- **Step 8:** Run the app
  - Should show map with a marker on Sydney Australia

More Maps Resources

- More code examples/demos at:
  - https://github.com/googlemaps/android-samples

- Add a map to your Android app (kotlin)
  - https://developers.google.com/codelabs/maps-platform/maps-platform-101-android#0



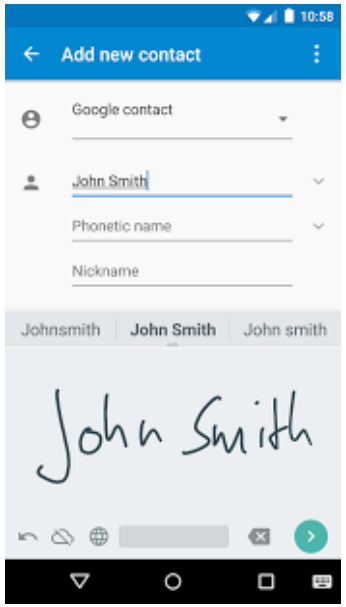


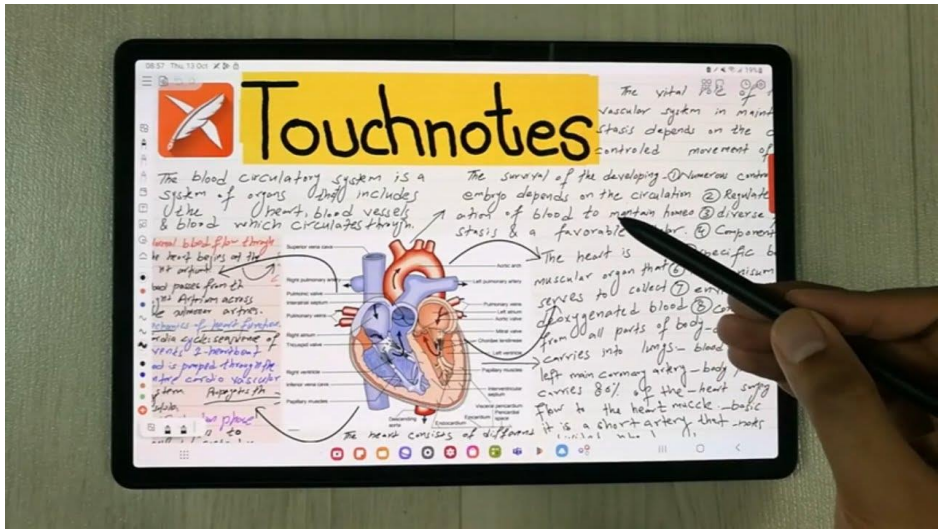
What other Android APIs may be useful for Mobile/ubicomp?

Touch Gestures & Stylus

https://developer.android.com/develop/ui/views/touch-and-input/gestures
https://developer.android.com/develop/ui/views/touch-and-input/stylus

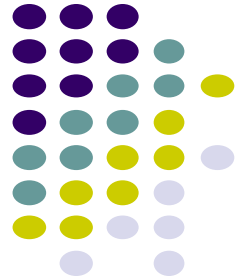
- Examples:
  - Common gestures: scrolling, flinging, double-tapping, swiping
  - Multi-touch, pinching
  - Stylus or handwriting on screen to search phone, contacts, etc.
  - Speed dial by writing first letters of contact's name
  - Drag and drop



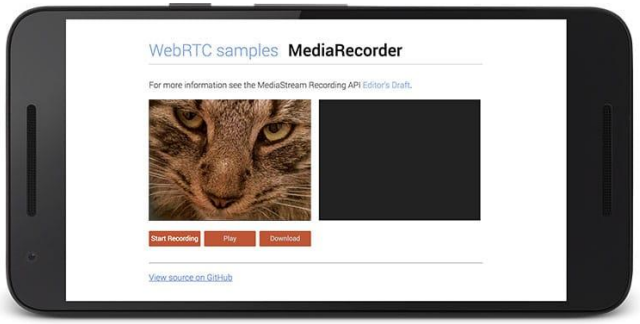


More MediaPlayer & GPGPU

https://developer.android.com/guide/topics/media/mediarecorder
https://android-developers.googleblog.com/2021/04/android-gpu-compute-going-forward.html
https://developer.android.com/ndk/guides/graphics/getting-started



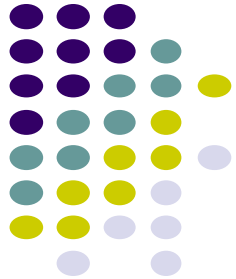
- MediaRecorder is used to **record** audio
  - Capture and encode various audio and video formats
  - Manipulate raw audio from microphone/audio hardware, PCM buffers
    - E.g. if you want to do audio signal processing, speaker recognition, etc
    - **Example:** process user's speech, detect emotion, nervousness?
  - Can playback recorded audio using MediaPlayer



- **GPGPU Programming using Vulkan and OpenGL**
  - Languages for computationally intensive tasks/GPGPU,
  - Use Phone's Graphics Processing Unit (GPU) for computational tasks
  - Useful for heavy duty tasks. E.g. image processing, computational photography, computer vision

Wireless Communication

https://developer.android.com/guide/topics/connectivity/bluetooth
https://developer.android.com/guide/topics/connectivity/wifi-scan
https://developer.android.com/guide/topics/connectivity/wifi-infrastructure



- Bluetooth
  - Discover, connect to nearby bluetooth devices
  - Communicating over Bluetooth
  - Exchange data with other devices
  - Example app: COVID contact tracing,
    Too Close for Too Long (< 6 ft. for > 15 mins)

- WiFi
  - Scan for WiFi hotspots
  - Monitor WiFi connectivity, Signal Strength (RSSI)
  - P2P: Connect using WiFi Direct (ad hoc networking)

Wireless Communication

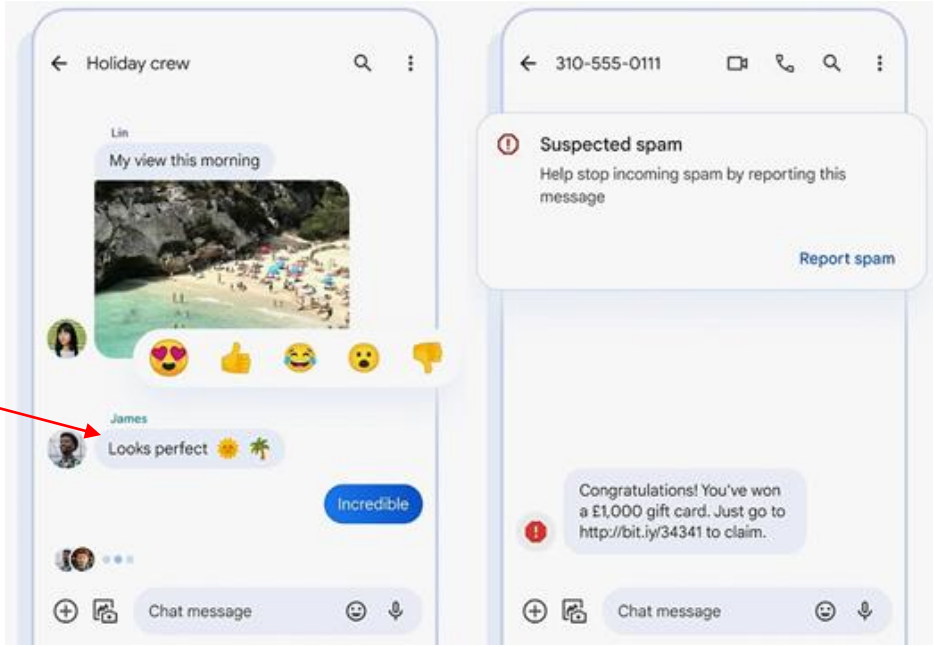
http://developer.android.com/guide/topics/connectivity/nfc/index.html

- NFC:
  - Contactless, transfer small amounts of data over short distances (4cm or less)
  - **Applications:** Share spotify playlists, Google wallet
  - **Android Pay**
    - Store debit, credit card on phone
    - Pay by tapping terminal

Telephony and SMS

http://developer.android.com/reference/android/telephony/package-summary.html
https://developer.android.com/reference/android/telephony/SmsManager

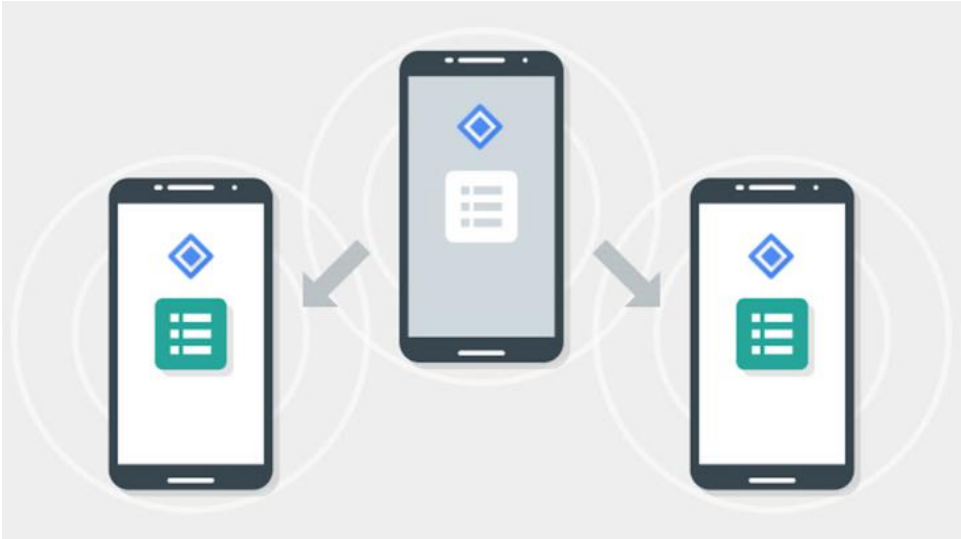
- **Telephony:**
  - Initiate phone calls from within app
  - Access dialer app, etc
- **SMS:**
  - Send/Receive SMS/MMS from app
  - Handle incoming SMS/MMS in app



Google Play Services: Nearby Connections API

https://developers.google.com/nearby/connections/overview

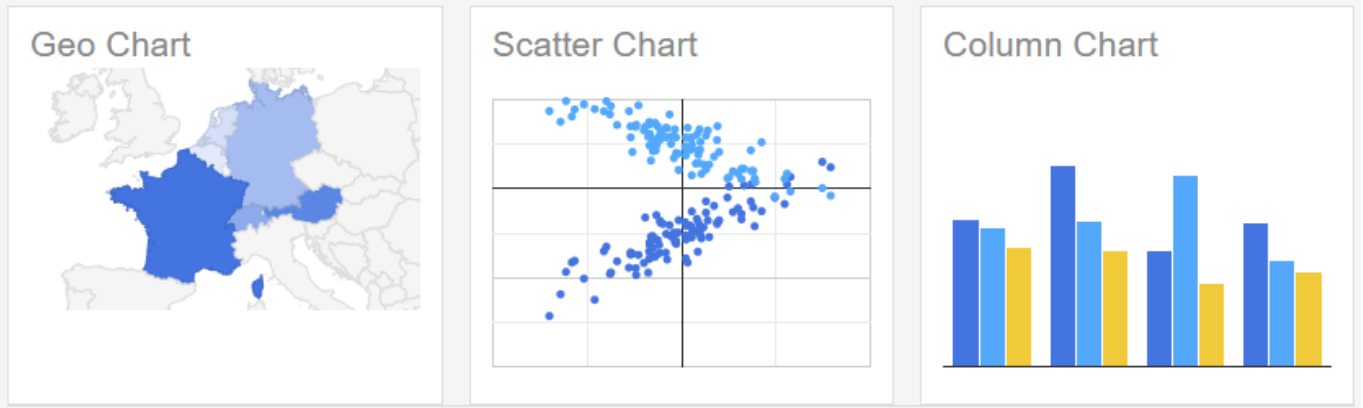
- Peer-to-peer networking API, without having to connect to Internet (i.e. over a LAN)
- Discover and directly communicate with other devices
- **Use cases:** Multiplayer gaming, shared virtual whiteboard
- One device serves as host, advertises
- Other devices can discover host, connect, disconnect



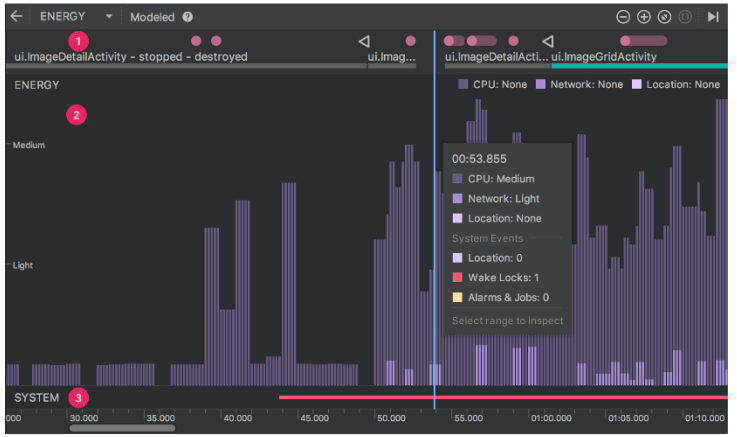
Visualizations/Charts and Energy Profiler

https://developer.android.com/studio/profile/energy-profiler
https://developers.google.com/chart

- **Google Charts:** Add charts to your app

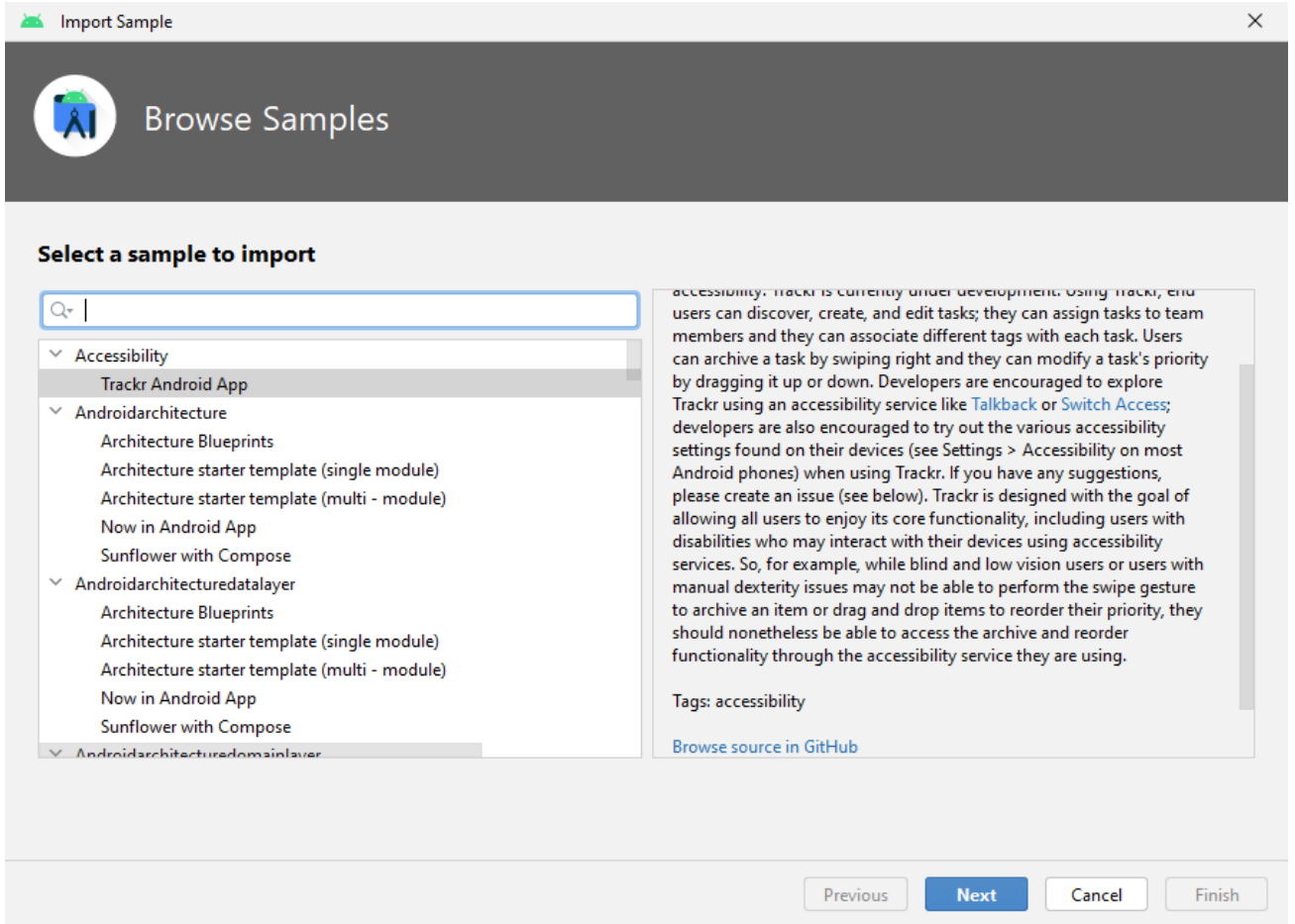


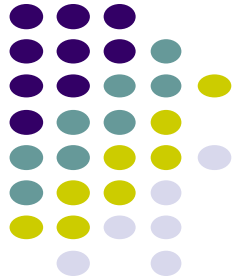
- **Energy Profiler:** Profile, visualize power usage of your app (CPU, network radio, GPS sensor, etc.)



Google Android Samples

- Android Studio has many sample programs
- Just need to import them
  - File -> New -> Import Sample…

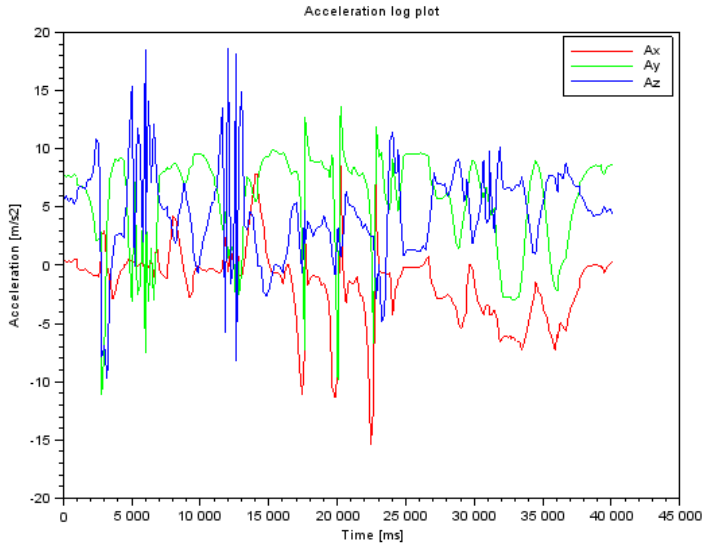




Introduction to Activity Recognition

Human Activity Recognition (HAR)

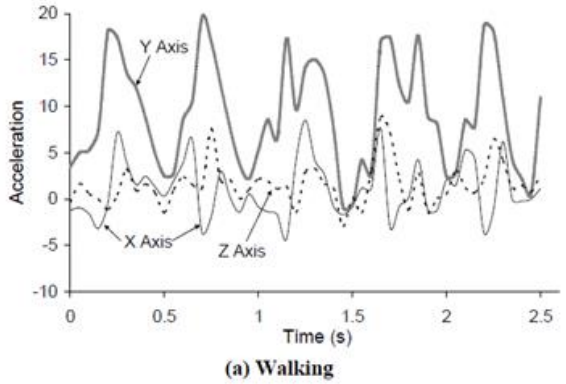
- **Goal:** Want app to detect what activity the user is doing?
- **HAR task:** which of these 6 activities is user doing?
  - Walking,
  - Jogging,
  - Ascending stairs,
  - Descending stairs,
  - Sitting,
  - Standing



- Typically, use machine learning classifers on user's accelerometer signals

Activity Recognition Overview

Smartphone Accelerometer data



(a) Walking

Machine Learning Classifier

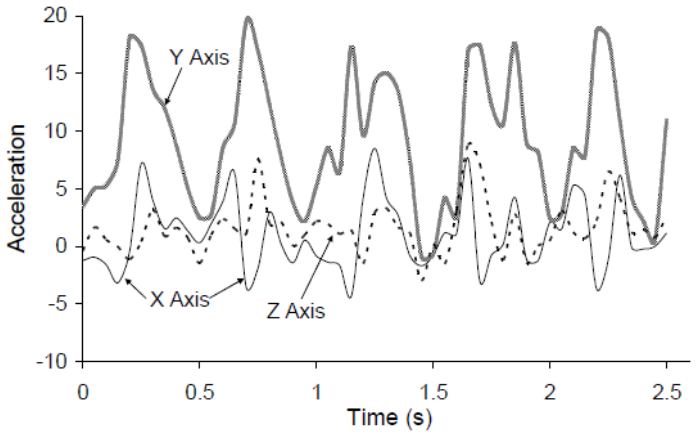
Classify Accelerometer data

Walking

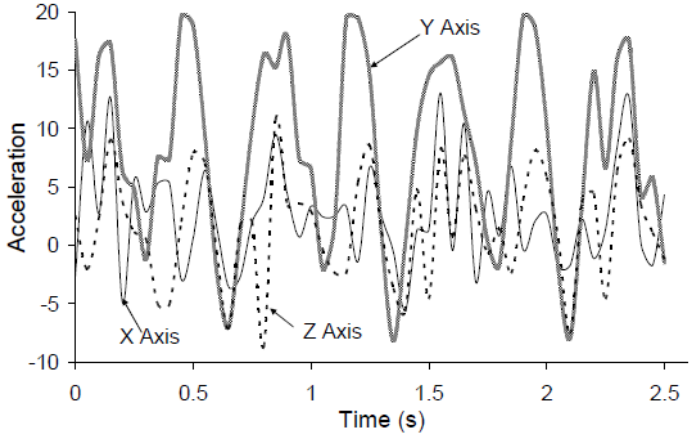
Running

Climbing Stairs

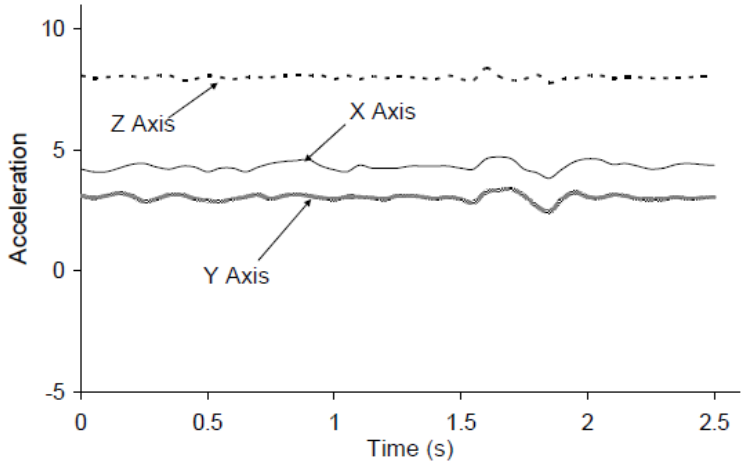
Accelerometer Patterns: Different for various Activities



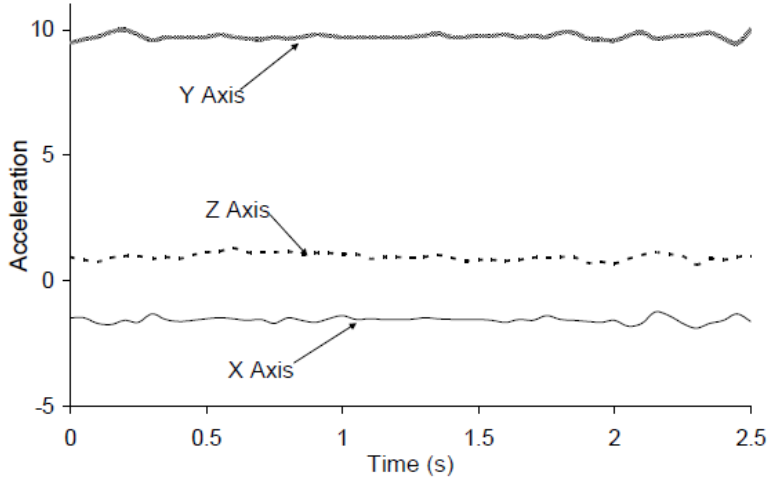
(a) Walking



(b) Jogging

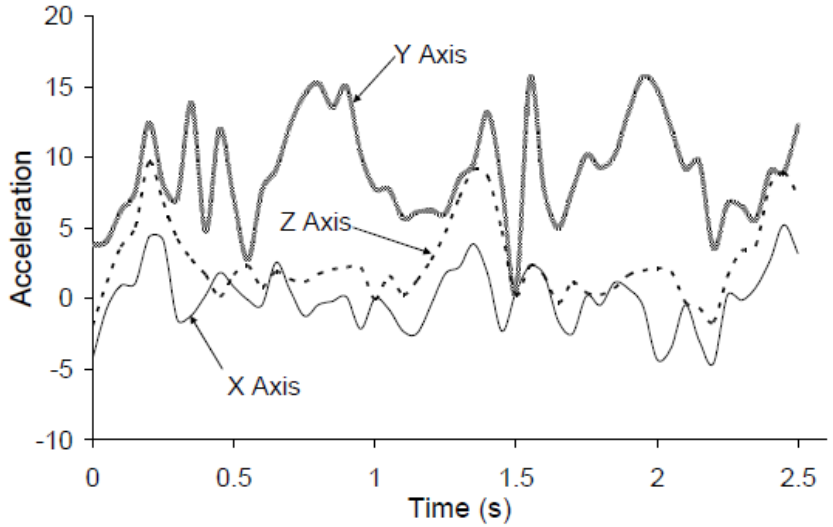


(e) Sitting

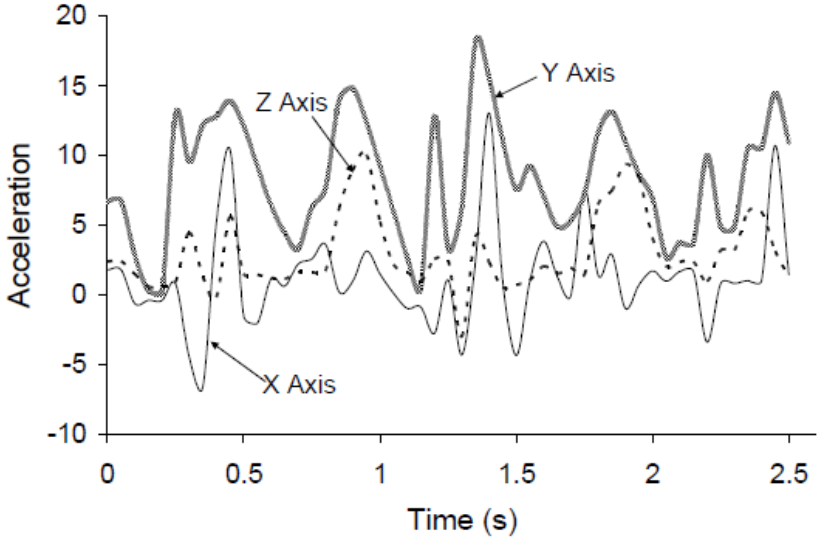


(f) Standing

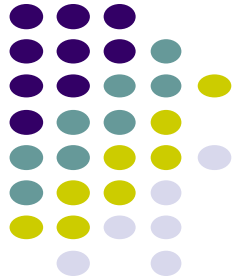
Example Accelerometer Data for Activities



(c) Ascending Stairs



(d) Descending Stairs

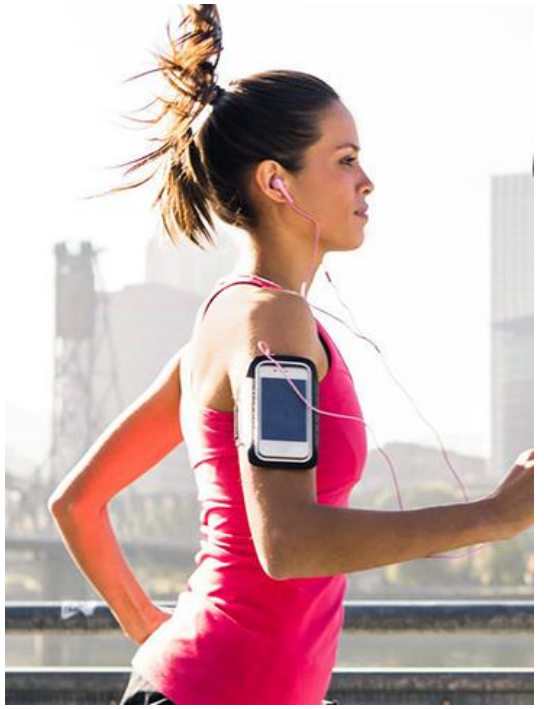


Applications of Activity Recognition

Applications of Activity Recognition (AR)

Ref: Lockhart *et al,* Applications of Mobile Activity recognition

- **Fitness Tracking:**
  - Physical activity type,
  - Distance travelled,
  - Calories burned
  - Stairs climbed,
  - Physical activity (duration + intensity)
  - Activity type logging + context
    - E.g. Ran 0.54 miles/hr faster during morning runs
  - Sleep tracking
  - Activity history



Note: AR algorithm can run on a range of devices (smartphones, smartwatches, wearables, e.g. fitbit)

Applications of Activity Recognition (AR)

Ref: Lockhart *et al,* Applications of Mobile Activity recognition

- **Health monitoring:** How **well** is patient performing activity?
- Pervasive, continuous monitoring in real-world!!
  - May also gather contextual information (e.g. what makes condition worse/better?)
- Show patient contexts that worsen condition => Change behavior
  - E.g. walking in narrow hallways worsens gait freeze

Parkinsons disease
Gait freezing

Question: What data would you collect to build PD gait classifier? From what types of subjects?

COPD, Walk tests in the wild

Applications of Activity Recognition

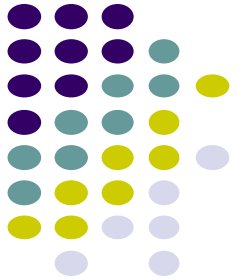
Ref: Lockhart *et al,* Applications of Mobile Activity recognition

- **Fall:** Leading cause of death for seniors
- **Fall detection:** Smartphone/watch, wearable detects senior who has fallen
  - Alert, text message, email, family or relative

Fall detection

Applications of Activity Recognition (AR)

Ref: Lockhart *et al,* Applications of Mobile Activity recognition



- **Context-Aware Behavior:**
  - In-meeting? => Phone switches to silent mode
  - Exercising? => Play song from playlist, use larger font sizes for text
  - Arrived at work? => download/refresh email

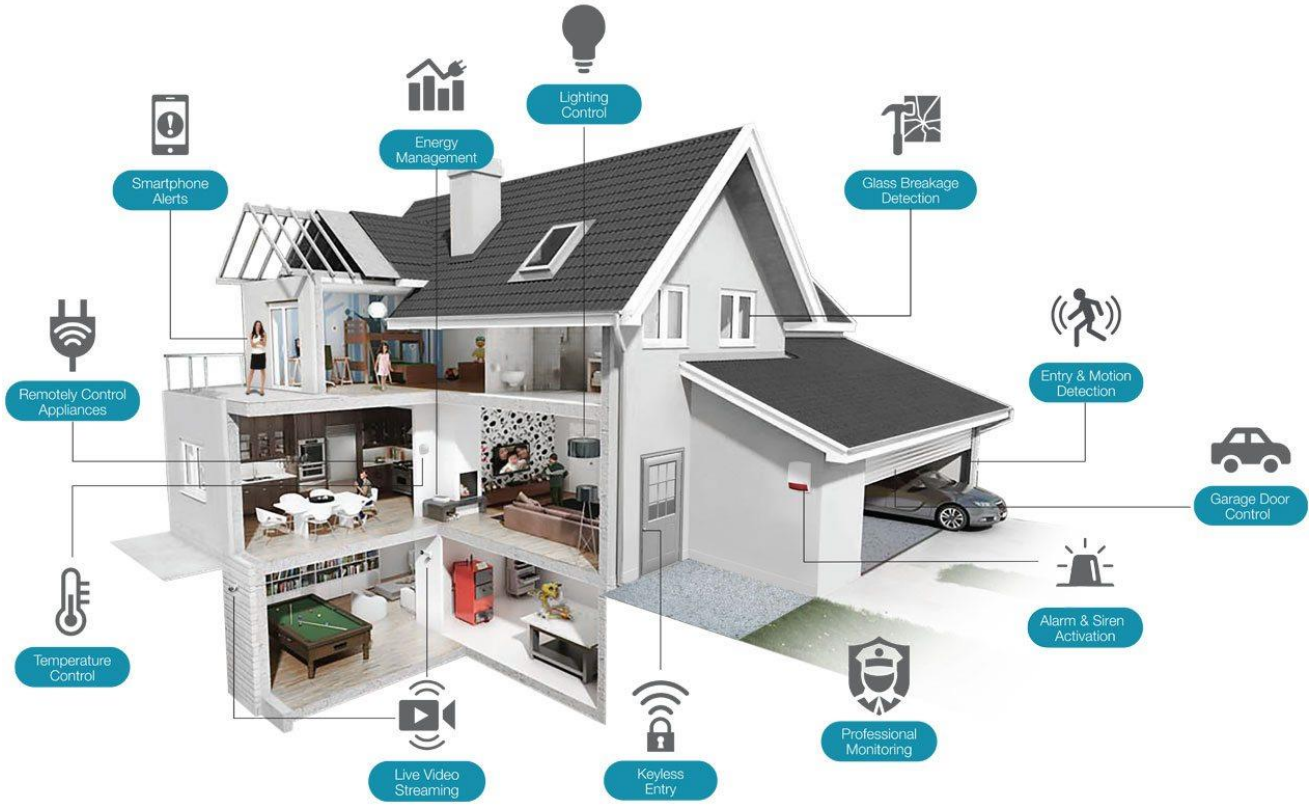
- Study found that messages delivered when transitioning between activities more likely to be better received

- **Adaptive Systems to Improve User Experience:**
  - Walking, running, riding bike? => Turn off Bluetooth, WiFi (save power)
  - Can increase battery life up to 5x

Applications of AR

Ref: Lockhart *et al,* Applications of Mobile Activity recognition

- **Smart home:**
  - Detect what activities people in the home are doing
  - **Why?** infer illness, wellness, patterns, intrusion (security), etc. E.g.
    - Detect user lying down a lot, not walking => ill?
    - Automatically turn on TV at time when user usually lies on the couch (Improved experience)



Applications of AR: 3rd Party Apps

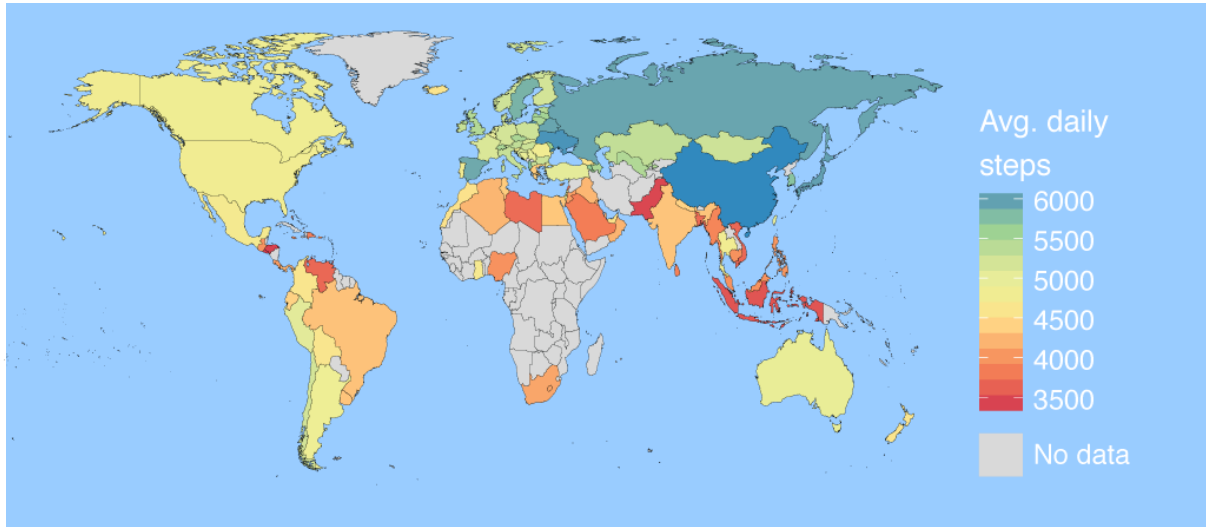
Ref: Lockhart *et al,* Applications of Mobile Activity recognition

- **Targeted Advertising:**
  - Deliver more relevant ads. E.g.
  - User runs a lot => Get exercise clothing ads
  - Goes to pizza places often + sits there => Get pizza ads

Applications of AR: 3rd Party Apps

Ref: Lockhart *et al,* Applications of Mobile Activity recognition

- **Research Platforms for Data Collection:**
  - E.g. public health officials want to know how much time various groups of people (e.g. students) spend sleeping, walking, exercising, etc
  - Mobile AR: inexpensive, automated data collection
  - E.g. Stanford Inequality project: Analyzed physical activity of 700k users in 111 countries using smartphone AR data
  - http://activityinequality.stanford.edu/



Applications of AR: 3rd Party Apps

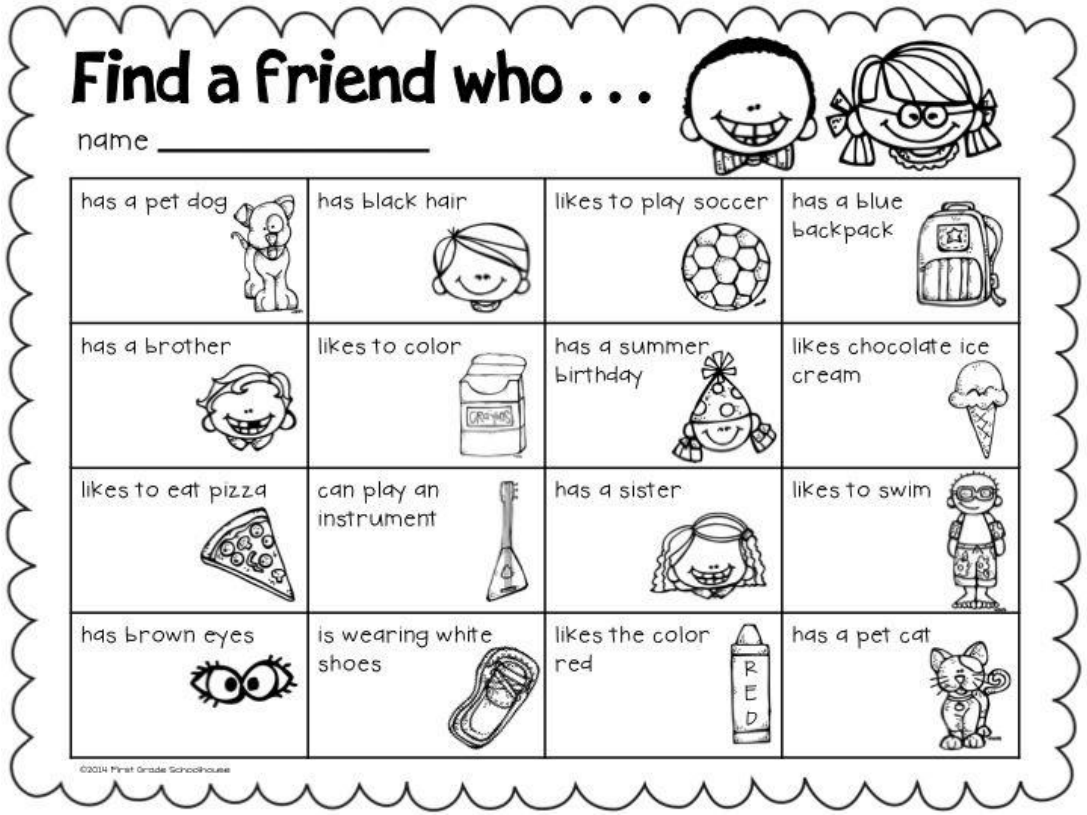
Ref: Lockhart *et al,* Applications of Mobile Activity recognition

- **Track, manage staff on-demand:**
  - At hospital, determine “availability of nurses”
  - E.g. nurses sitting? Assign them to incoming patients/surgeries/cases

Applications of AR: Social Networking

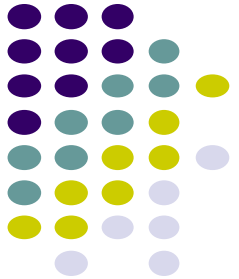
Ref: Lockhart *et al,* Applications of Mobile Activity recognition

- **Activity-Based Social Networking:**
  - Automatically connect users who do same activities + live close together



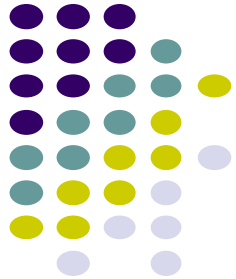
Applications of AR: Social Networking

Ref: Lockhart *et al,* Applications of Mobile Activity recognition



- **Activity-Based Place Tagging:**
  - Automatically “popular” places where users perform same activity
  - E.g. Park street is popular for runners (activity-based maps)

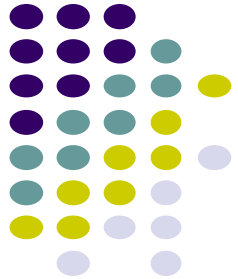
- **Automatic Status updates:**
  - E.g. Bob is sleeping
  - Tracy is jogging along Broadway with track team
  - Privacy/security concerns => Different Levels of details for different friends



Activity Recognition Using Google API

Google Activity Recognition Transition API

https://developer.android.com/guide/topics/location/transitions

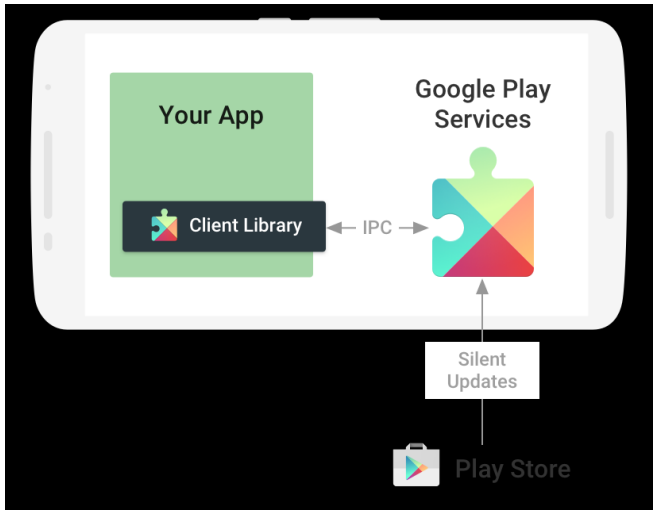


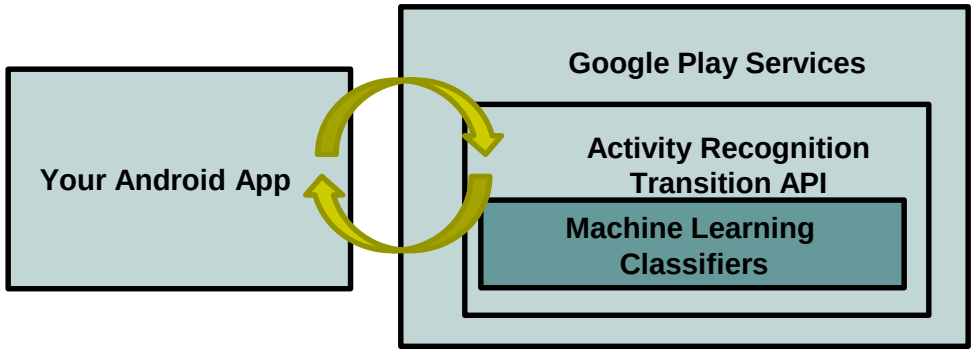
- API to detect smartphone user's current activity
- Why? App can adapt it's behavior based on user behavior
  - **E.g.** If user is driving, don't send notifications
- Programmable, can be used by your Android app
- Currently AR API detects when user starts or ends 5 activities:
  - In vehicle
  - On Bicycle
  - Running
  - Walking
  - Still

Google Activity Recognition TransitionAPI

https://developer.android.com/guide/topics/location/transitions

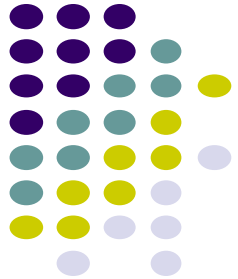
- Deployed as part of Google Play Services





Google Activity Recognition Transition API

https://developer.android.com/guide/topics/location/transitions



- First, set up your app to use Google Play services, see
  - https://developer.android.com/guide/topics/location/transitions

- Add permissions necessary for Activity Recognition

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="com.example.myapp">

  <uses-permission android:name="com.google.android.gms.permission.ACTIVITY_RECOGNITION" />
  …
</manifest>
```

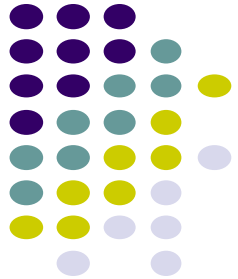
Google Activity Recognition Transition API

https://developer.android.com/guide/topics/location/transitions

- Need to register for Activity Transition updates, which requires implementing:
  - **ActivityTransitionRequest** object: Specifies activity type (e.g. walking), and transitions (enter, exit)
  - **PendingIntent** callback where app receives notifications
- Creating **ActivityTransitionRequest** object requires creating list of **ActivityTransition** objects
- **ActivityTransition** object includes:
  - Activity type (represented by **DetectedActivity** class)
    - IN_VEHICLE
    - ON_BICYCLE
    - RUNNING
    - STILL
    - WALKING
  - Transition type:
    - ACTIVITY_TRANSITION_ENTER
    - ACTIVITY_TRANSITION_EXIT

Google Activity Recognition Transition API

https://developer.android.com/guide/topics/location/transitions



- Create list of **ActivityTransition** objects

```
val transitions = mutableListOf<ActivityTransition>()

transitions +=
        ActivityTransition.Builder()
          .setActivityType(DetectedActivity.IN_VEHICLE)
          .setActivityTransition(ActivityTransition.ACTIVITY_TRANSITION_ENTER)
          .build()

transitions +=
        ActivityTransition.Builder()
          .setActivityType(DetectedActivity.IN_VEHICLE)
          .setActivityTransition(ActivityTransition.ACTIVITY_TRANSITION_EXIT)
          .build()

transitions +=
        ActivityTransition.Builder()
          .setActivityType(DetectedActivity.WALKING)
          .setActivityTransition(ActivityTransition.ACTIVITY_TRANSITION_EXIT)
          .build()
```

App notified when user starts being in vehicle

App notified when user stops being in vehicle

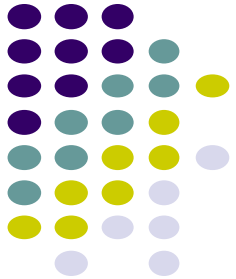
App notified when user stops walking

- Pass list of **ActivityTransitions** to **ActivityTransitionRequest** class

```
val request = ActivityTransitionRequest(transitions)
```

Google Activity Recognition Transition API

https://developer.android.com/guide/topics/location/transitions



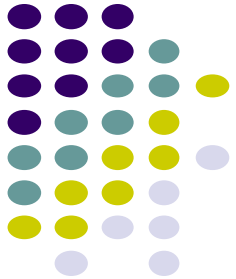
- **PendingIntent:** wrapper around **Intent** object,
  - Grants permission to foreign app to send intents to an app.
  - E.g. Android AR API can send our app pendingIntent
- Register for activity transition updates by
  - Passing **ActivityTransitionRequest** and **PendingIntent** object to **requestActivityTransitionUpdates( )** method
  - Returns **Task** object

```
// myPendingIntent is the instance of PendingIntent where the app receives callbacks.
val task = ActivityRecognition.getClient(context)
        .requestActivityTransitionUpdates(request, myPendingIntent)

task.addOnSuccessListener {
    // Handle success
}

task.addOnFailureListener { e: Exception ->
    // Handle error
}
```

Check return values for success/failure and handle

Google Activity Recognition Transition API

https://developer.android.com/guide/topics/location/transitions

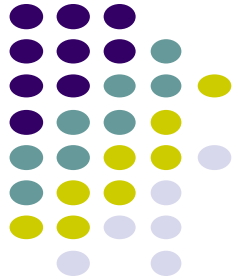


- After registering for activity updates, app receives notifications in **PendingIntent**
- When requested activity transition occurs, app receives **Intent** callback
- An **ActivityTransitionResult** object can be extracted from the **Intent**
  - Contains list of **ActivityTransitionEvent** objects
  - E.g. <IN_VEHICLE, ACTIVITY_TRANSITION_ENTER>
- Can create Broadcast receiver, receive list of activity transitions in its **onReceive( )** method

```
override fun onReceive(context: Context, intent: Intent) {
    if (ActivityTransitionResult.hasResult(intent)) {
        val result = ActivityTransitionResult.extractResult(intent)!!
        for (event in result.transitionEvents) {
            // chronological sequence of events....
        }
    }
}
```

Google Activity Recognition Transition API

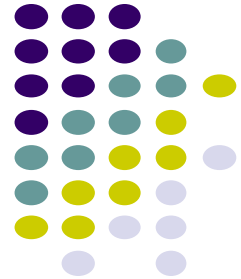
https://developer.android.com/guide/topics/location/transitions



- When done, de-register from receiving activity transition updates by calling **removeActivityTransitionUpdates( )** method

```
// myPendingIntent is the instance of PendingIntent where the app receives callbacks.
val task = ActivityRecognition.getClient(context)
        .removeActivityTransitionUpdates(myPendingIntent)

task.addOnSuccessListener {
    myPendingIntent.cancel()
}

task.addOnFailureListener { e: Exception ->
    Log.e("MYCOMPONENT", e.message)
}
```

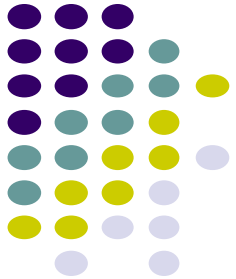
Pass in **PendingIntent**



Android Awareness API

Awareness API

https://developers.google.com/awareness/overview



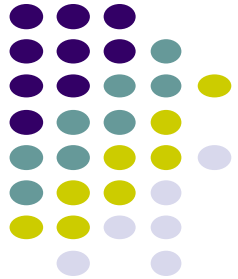
- Single Android API for context awareness released in 2016
- Combines some APIs already covered (E.g. Activity, Location)

Context type	Example
Time	Current local time
Location	Latitude and longitude
Activity	Detected user activity, like walking, running, or biking
Beacons	Nearby beacons that match the specified namespace
Headphones	Status of whether headphones are plugged in, or not

- **Note:** Previously included Place and Weather information, now deprecated

Awareness API

https://developers.google.com/awareness/overview
https://developers.google.com/awareness/android-api/fence-api-overview

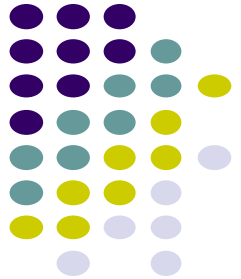


- **Snapshot API:**
  - Used to query user's current context (Activity, location, etc)
  - System caches values
  - Optimized for battery and power consumption

- **Fences API:**
  - Used to set combination of conditions to trigger events
  - E.g. Notify app if user
    - Enters a geoFence & Activity is running
    - Walking & headphones plugged in

- More information at Android Awareness API website

References



- Android Nerd Ranch, 5th edition
- Google Android Tutorials