UNIVERSITY OF
PLYMOUTH

COMP2000: Software engineering 2

Week 9: Location awareness

Outline

- Location access
- Geofencing

Location feature

- One of the unique features of mobile applications is location awareness.
- Adding location awareness to your app offers users a more contextual experience.
- The location APIs available in Google Play services facilitate adding location awareness to your app with automated location tracking, wrong-side-of-the-street detection, geofencing, and activity recognition.

Request permission

- To protect user privacy, apps that use location services must request location permissions.

There are two types of location access: Foreground and background permission.

Each permission has a combination of the following characteristics:

- **Category**: Either foreground location or background location.
- **Accuracy**: Either precise location or approximate location.

Foreground location

If your app contains a feature that shares or receives location information only once, or for a defined amount of time, then that feature requires foreground location access.

Examples:

- Within a navigation app, a feature allows users to get turn-by-turn directions.
- Within a messaging app, a feature allows users to share their current location with another user.

Manifest file

```
<!-- Recommended for Android 9 (API level 28) and lower. -->
   <!-- Required for Android 10 (API level 29) and higher. -->
   <service
      android:name="MainService"
      android:foregroundServiceType="location">
   <!-- Any inner elements would go here. -->
</service>
```

```
<!-- Always include this permission -->
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />

<!-- Include only if your app benefits from precise location access. -->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Create location services client

- First: add dependencies in the Gradle file:

```
implementation 'com.google.android.gms:play-services-location:18.0.0'
```

Background location

An app requires background location access if a feature within the app constantly shares location with other users or uses the Geofencing API.

Examples:

•Within a family location sharing app, a feature allows users to continuously share location with family members.

•Within an IoT app, a feature allows users to configure their home devices such that they turn off when the user leaves their home and turn back on when the user returns home.

```
<!-- Required only when requesting background location access on
   Android 10 (API level 29) and higher. -->
<uses-permission
android:name="android.permission.ACCESS_BACKGROUND_LOCATION"
/>
```

Any other examples of Foreground or background location access?

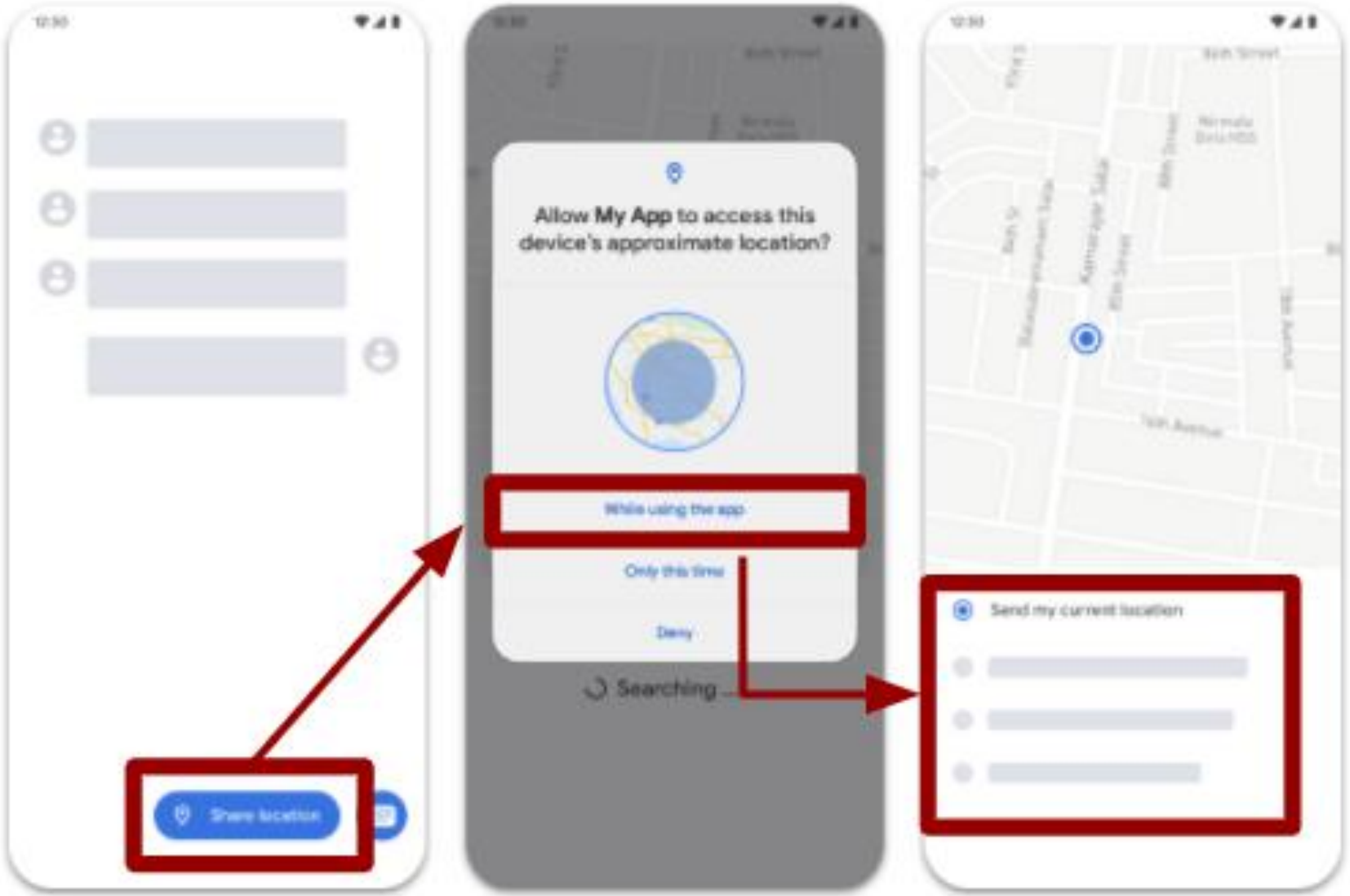


Figure 1. Location-sharing feature that requires foreground location access. The feature is enabled if the user selects **Allow only while using the app**.

User choice affects permission grants

	Precise	Approximate
While using the app	`ACCESS_FINE_LOCATION` and `ACCESS_COARSE_LOCATION`	`ACCESS_COARSE_LOCATION`
Only this time	`ACCESS_FINE_LOCATION` and `ACCESS_COARSE_LOCATION`	`ACCESS_COARSE_LOCATION`
Deny	No location permissions	No location permissions

Check permission

```
protected void checkPermission(){
   if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)
!= PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
ActivityCompat.requestPermissions(this, new String[]{
            Manifest.permission.ACCESS_COARSE_LOCATION,
            Manifest.permission.ACCESS_FINE_LOCATION}, REQUEST_CHECK_SETTINGS);

   }
}
```

Get the last known location

- Using the Google Play services location APIs, your app can request the last known location of the user's device.

- Use the fused location provider to retrieve the device's last known location.

```
private FusedLocationProviderClient fusedLocationClient;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    //....
    fusedLocationClient = LocationServices.getFusedLocationProviderClient(this);

}
```

Get the last known location

```
fusedLocationClient.getLastLocation()
        .addOnSuccessListener(this, new OnSuccessListener<Location>() {
            @Override
            public void onSuccess(Location location) {
                // Got last known location. In some rare situations this can be null.
                if (location != null) {
                    // Logic to handle location object
                }
            }
        });
```

Get the latitude and longitude

```
Log.d("Latitude", String.valueOf(location.getLatitude()));
Log.d("Longitude", String.valueOf(location.getLongitude()));
```

```
txt.setText(String.valueOf(location.getLatitude())+ "  "+ String.valueOf(location.getLongitude()));
```

In case of failure:

```
fusedLocationClient.getLastLocation().addOnFailureListener(this, new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception e) {
        Log.d("Exception", "Location is not working");
        Toast.makeText(MainActivity.this, "Location failed", Toast.LENGTH_SHORT).show();

    }
});
```

Set up a location request

```
protected void createLocationRequest() {
    LocationRequest locationRequest = LocationRequest.create();
    locationRequest.setInterval(10000);
    locationRequest.setFastestInterval(5000);

locationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
}
```

Get current location settings

```
LocationSettingsRequest.Builder builder = new LocationSettingsRequest.Builder()
        .addLocationRequest(locationRequest);
```

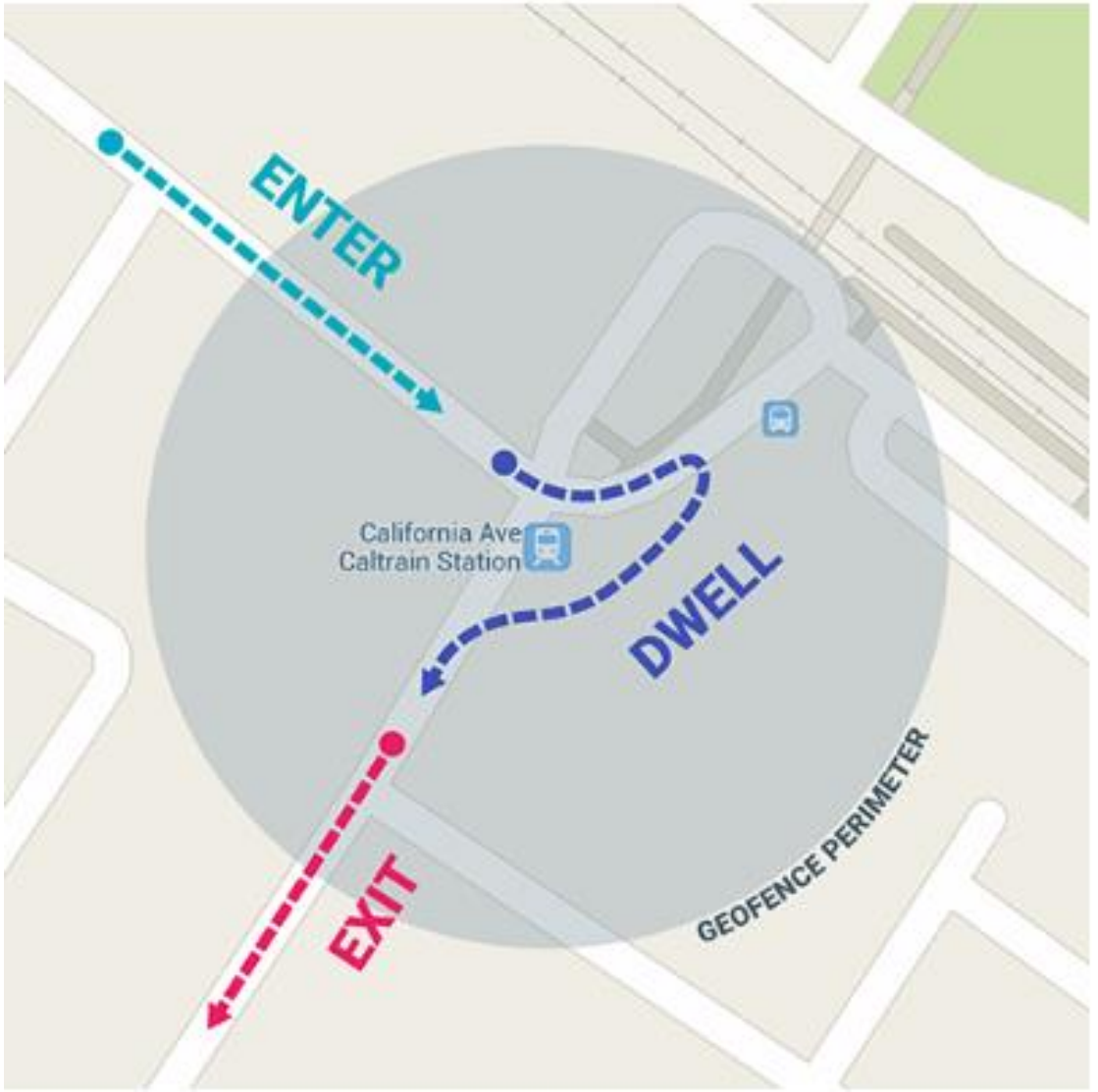
Check if the location access is enabled:

```
private boolean isLocationEnabled() {
    LocationManager locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
    return
locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER) ||
locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER);
}
```

Examples of using background Location access

- Geofence technology

Geo-fencing


ENTER
California Ave
Caltrain Station
DWELL
EXIT
GEOFENCE PERIMETER

Create an instance of the Geofencing client

```
private GeofencingClient geofencingClient;
```

Create geofence objects

To create a geofence object you need to use Geofence.Builder to and setting the desired radius, duration, and transition types for the geofence.

Example to populate a list object:

```
geofencingClient = LocationServices.getGeofencingClient(this);
Map.Entry<String, LatLng> entry = null;
geofenceList.add(new Geofence.Builder()
        // Set the request ID of the geofence. This is a string to identify this geofence.
        .setRequestId(entry.getKey())
        .setCircularRegion(
                entry.getValue().latitude,
                entry.getValue().longitude,
                Constants.GEOFENCE_RADIUS_IN_METERS
        )

.setExpirationDuration(Constants.GEOFENCE_EXPIRATION_IN_MILLISECONDS)
        .setTransitionTypes(Geofence.GEOFENCE_TRANSITION_ENTER |
                Geofence.GEOFENCE_TRANSITION_EXIT)
        .build());
```

Specify geofences and initial triggers

```
private GeofencingRequest getGeofencingRequest() {
    GeofencingRequest.Builder builder = new GeofencingRequest.Builder();
    builder.setInitialTrigger(GeofencingRequest.INITIAL_TRIGGER_ENTER);
    builder.addGeofences(geofenceList);
    return builder.build();
```

Define a broadcast receiver service for geofence transitions using Intent

```
private PendingIntent getGeofencePendingIntent() {
    // Reuse the PendingIntent if we already have it.
    if (geofencePendingIntent != null) {
        return geofencePendingIntent;
    }
    Intent intent = new Intent(this, GeofenceTransitionsIntentService.class);
    // We use FLAG_UPDATE_CURRENT so that we get the same pending intent back when
    // calling addGeofences() and removeGeofences().
    geofencePendingIntent = PendingIntent.getBroadcast(this, 0, intent, PendingIntent.
            FLAG_UPDATE_CURRENT);
    return geofencePendingIntent;
}
```

To add geofences, use the GeofencingClient.addGeofences() method.
Provide the GeofencingRequest object, and the PendingIntent. See the following example:

```
geofencingClient.addGeofences(getGeofencingRequest(), getGeofencePendingIntent())
        .addOnSuccessListener(this, new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void aVoid) {
                // Geofences added
                // ...
            }
        })
        .addOnFailureListener(this, new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                // Failed to add geofences
                // ...
            }
        });
```

Handle geofence transitions

When Location Services detects that the user has entered or exited a geofence, it sends out the Intent contained in the PendingIntent you included in the request to add geofences.

A broadcast receiver like GeofenceBroadcastReceiver notices that the Intent was invoked and can then obtain the geofencing event from the intent, determine the type of Geofence transition(s), and determine which of the defined geofences was triggered.

The broadcast receiver can direct an app to start performing background work or, if desired, send a notification as output.

Stop the service

- Stopping geofence monitoring when it is no longer needed or desired can help save battery power and CPU cycles on the device.

- To stop the service, you need to use GeofencingClient.removeGeofences() method.

- **Try to do it yourself.**
- **Then try to put everything together for the lab session.**

- **More information you could fine here: https://developer.android.com/develop/sensors-and-location/location/geofencing**

- Thank you