



Google Maps

# Google Maps Platform Key Services

- **Maps:**

- Apps can integrate customized and interactive maps, satellite imagery and Street View imagery



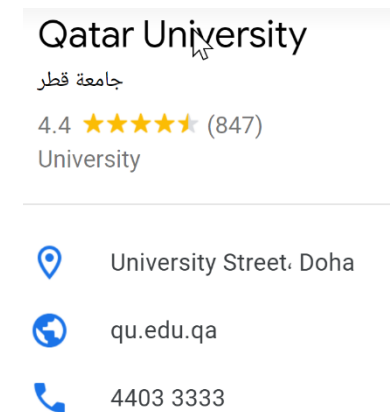
- **Routes:**

- Allow users to find the best route to get from A to Z using public transport, driving, biking, or walking.
- Compute travel times and distances
- Real-time traffic updates about the selected route



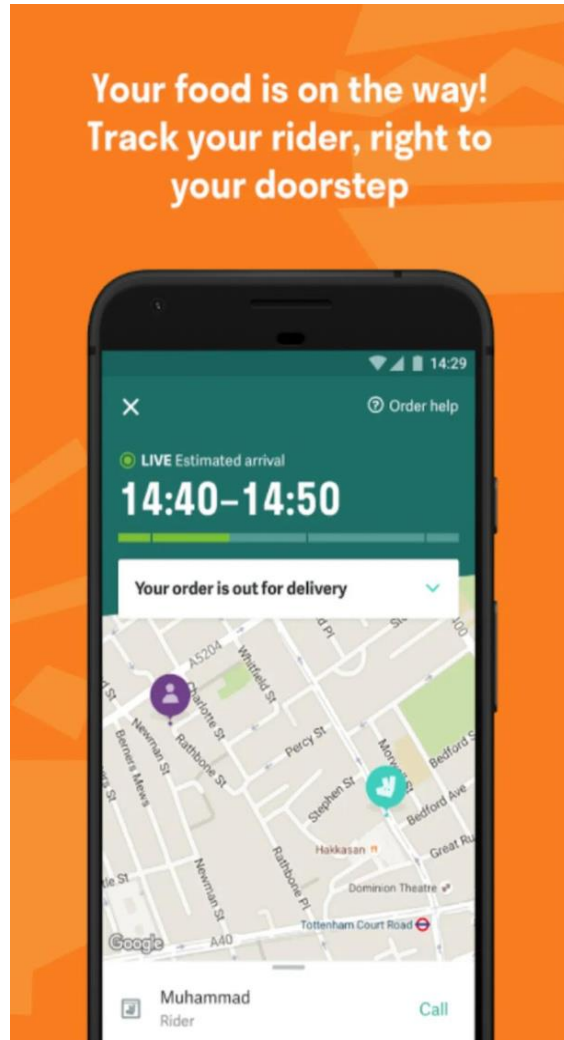
- **Places:**

- Users can search details about million **points of interest** around the world including place names, addresses, images, contact information and reviews

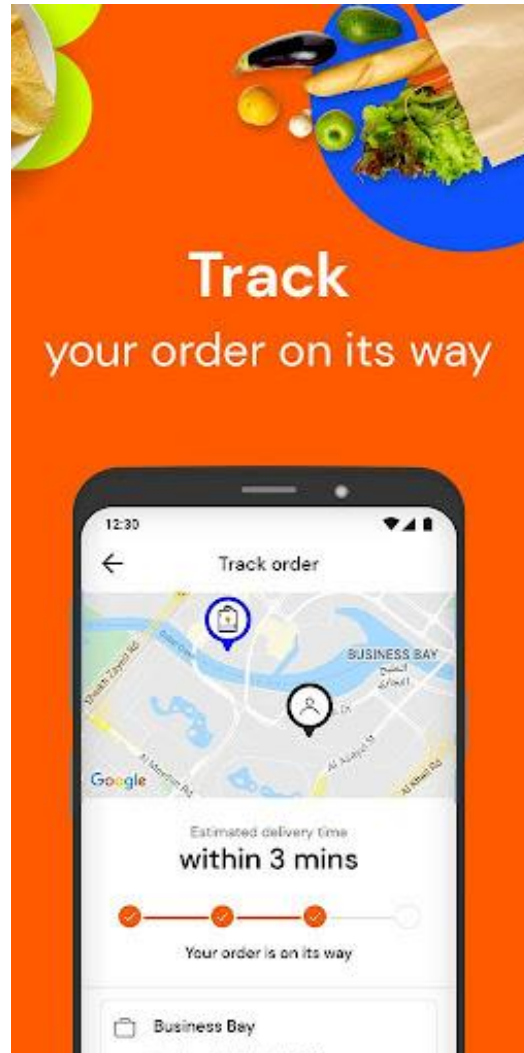


# What's driving growth of Map Apps?

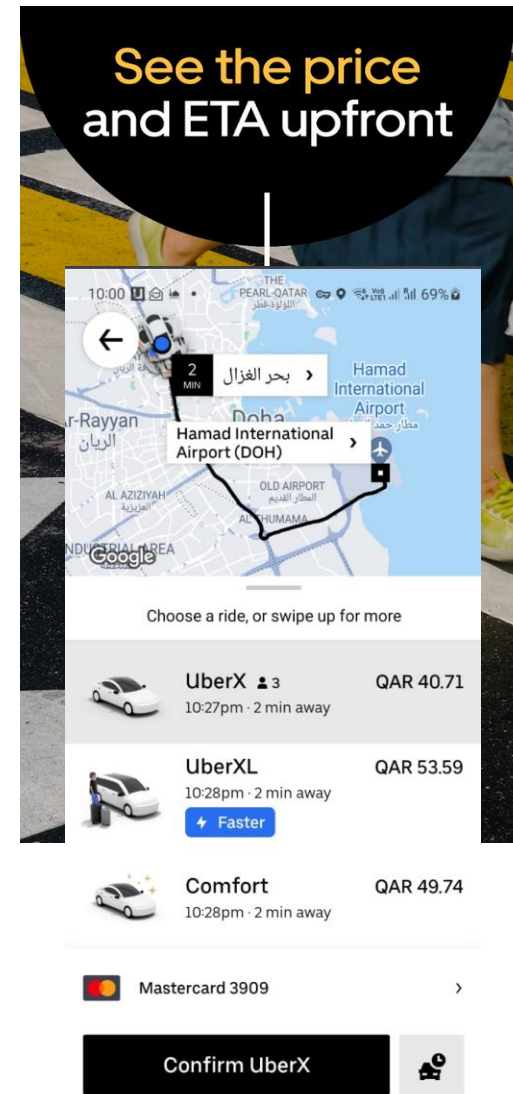
## Grocery delivery apps



## Food delivery apps



## Ride hailing apps



# Dependencies

- Add these dependencies to build.gradle

```
val mapsComposeVersion = "4.3.0"
implementation("com.google.maps.android:maps-compose:$mapsComposeVersion")

// Optionally, you can include the Compose utils library for Clustering,
// Street View metadata checks, etc.
implementation("com.google.maps.android:maps-compose-utils:$mapsComposeVersion")

// Optionally, you can include the widgets library for ScaleBar, etc.
implementation("com.google.maps.android:maps-compose-widgets:$mapsComposeVersion")
```

- Need to add Google Maps API key to *AndroidManifest.xml* file

More details on how to get the API Key

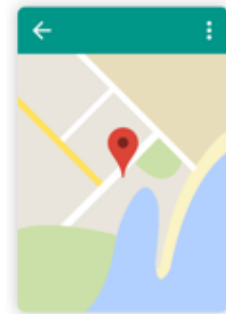
<https://developers.google.com/maps/documentation/android-sdk/get-api-key>

# Typical Programming Tasks in Location-aware App

- Visualise data in a **custom map**
- Get the device **geolocation** (latitude & longitude)
- **Geocoding**: finding the GPS coordinates of an address
  - E.g., what are coordinates of Qatar University
- **Reverse Geocoding**: finding the address of a GPS coordinates
  - E.g., what is the address at
- **Location tracking** as the user moves
  - Uber track current location during the ride
- **Geofencing**: trigger an action/notification when the device is in area of interest
  - E.g., switch on the corridor light when the user approaches the area of their home

# Display a Map

- Use **GoogleMap** component to display an interactive Google Maps
- The displayed map can be **customized** such as:
  - Add marker
  - Add overlay (e.g., image over the map)
  - Change the zoom level
  - Handle events such as Point of Interest (PoI) click event



```
val quPosition = LatLng(25.377, 51.491)
val zoomLevel = 20f // Buildings

val cameraPositionState =
    rememberCameraPositionState {
        position = CameraPosition.
            fromLatLngZoom(quPosition,
                zoomLevel)
    }

GoogleMap(
    modifier = Modifier.fillMaxSize(),
    cameraPositionState = cameraPositionState
) {
    Marker(
        state = MarkerState(
            position = quLocation),
        title = "QU",
        snippet = "Qatar University"
    )
}
```

# Zoom to a Location

- **GoogleMap** component a `cameraPositionState` parameter to know where GoogleMaps should look and the zoom level
  - Look at a particular **geo coordinates** and change the zoom level
- Zoom level values:
  - 1: World
  - 5: Continent
  - 10: City
  - 15: Streets
  - 20: Buildings

```
val quPosition = LatLng(25.377, 51.491)
val zoomLevel = 20f // Buildings
val cameraPositionState =
    rememberCameraPositionState {
        position = CameraPosition.fromLatLngZoom(
            quPosition, zoomLevel)
    }

GoogleMap(
    cameraPositionState,
    modifier = Modifier.fillMaxSize()
) { }
```



# Add Marker

- Marker identify a location on the map at a particular geo coordinates
  - Use the **Marker** composable to add a marker containing the coordinates on the map, with a title and small description snippet
  - When the marker is clicked an **info window** displays the marker's title and snippet text

*// A Snippet is a text displayed below the title*

```
val snippetText = "Lat: ${quLocation.latitude},  
Long: ${quLocation.longitude}"
```

```
Marker(  
    state = MarkerState(position = quLocation),  
    title = "Qatar University",  
    snippet = snippetText  
)
```





# Marker Custom Window

- Use **MarkerInfoWindowContent**
- Here we add a column with two text and an Image
- Useful if you want to show more details about the marker



```
MarkerInfoWindow(  
    state = MarkerState(position = quLocation),  
) {  
  
    Column {  
        Image(  
            painter = painterResource(  
                id = R.drawable.img_qu),  
            contentDescription = "QU"),  
        Text(text = "My Uni جامعتي" )  
        Text(text = snippetText)  
    }  
}
```

# Map Customization

- Configuring the map can be done by passing a **MapProperties** object into the GoogleMap
- For UI-related configurations, use **MapUiSettings**

```
val mapProperties by remember {
    mutableStateOf(
        // Map type could be NORMAL, HYBRID, SATELLITE, TERRAIN
        MapProperties(mapType = MapType.HYBRID, isMyLocationEnabled = true)
    )
}
val mapUiSettings by remember {
    mutableStateOf(
        MapUiSettings(mapToolbarEnabled = true, zoomControlsEnabled = true)
    )
}
GoogleMap(
    cameraPositionState = cameraPositionState,
    properties = mapProperties, uiSettings = mapUiSettings,
    modifier = Modifier.fillMaxSize()
) {}
```

# Markers Cluster

- The marker clustering utility helps manage multiple markers at different zoom levels. When a user views the map at a high zoom level, the individual markers show on the map. When the user zooms out, the markers gather together into clusters, to make viewing the map easier.

```
val items = remember { mutableStateListOf<MapClusterItem>() } Out Of Date | 1 ✕ 1 ^
LaunchedEffect(Unit) { this: CoroutineScope
    items.add(MapClusterItem(quPosition, itemTitle: "Qatar University"))
    items.add(MapClusterItem(hamadAirportPosition, itemTitle: "Hamad International Airport"))
    items.add(MapClusterItem(islamicMuseumPosition, itemTitle: "Museum of Islamic Art"))
    items.add(MapClusterItem(hamadStadiumPosition, itemTitle: "Hamad bin Khalifa Stadium"))
    items.add(MapClusterItem(LatLng(latitude: 25.420738, longitude: 51.490154), itemTitle: "Lu
    items.add(MapClusterItem(LatLng(latitude: 25.3607, longitude: 51.4811), itemTitle: "Univer
})
MapMarkersClustering(items = items)
}

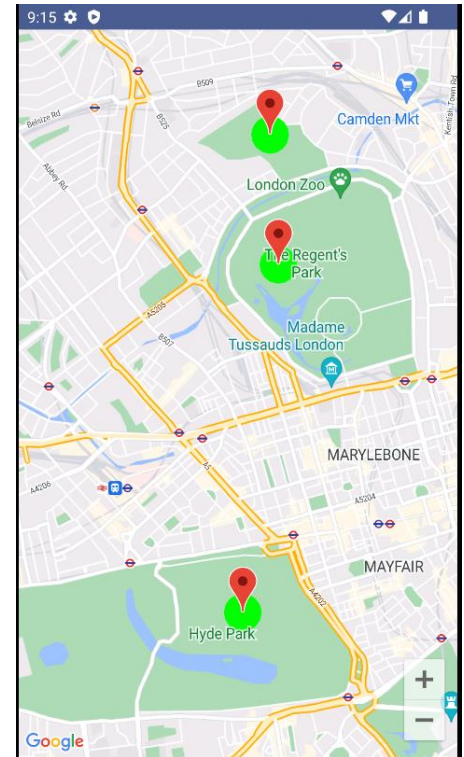
@Composable
fun MapMarkersClustering(items: List<MapClusterItem>) {
    GoogleMap(
        modifier = Modifier.fillMaxSize(),
        cameraPositionState = rememberCameraPositionState { this: CameraPositionState
            position = CameraPosition.fromLatLngZoom(quPosition, zoom: 10f)
        }
    ) {
        Clustering(
            items = items,|
```



# Drawing Shapes

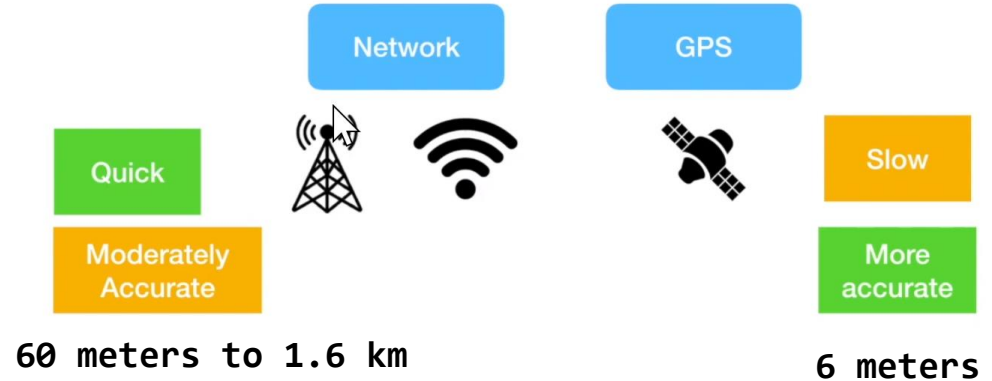
```
Box(contentAlignment = Alignment.Center) {
    GoogleMap(
        modifier = Modifier.fillMaxSize(),
        cameraPositionState = cameraPositionState
    ) {
        Clustering(items = parkMarkers,
            onClusterClick = {
                // Handle when the cluster is tapped
                false
            }, onClusterItemClick = { marker ->
                // Handle when a marker in the cluster is tapped
                selectedMarker = marker
                false
            })

        parkMarkers.forEach {
            Circle(
                center = it.position,
                radius = 120.0,
                fillColor = Color.Green,
                strokeColor = Color.Green
            )
        }
    }
}
```



# Get User Location

- Request last known location of the user's device
  - Location is determined by the **LocationServices** using WiFi & Cellular Tower and/or GPS (Global Positioning System)



```
val fusedLocationClient =  
    LocationServices.getFusedLocationProviderClient(appContext)  
val lastLocation = fusedLocationClient.lastLocation.await()  
lastLocation?.let {  
    val currentLocation = "Lat: ${it.latitude} & Long: ${it.longitude}"  
    println(">> Debug: $currentLocation")  
}  
}
```

# Request Location Permission

- At runtime must ask for the permission to access the device's location using

```
rememberLauncherForActivityResult(  
    ActivityResultContracts.RequestPermission())
```

*// Register request permission callback, which handles the user's response to the system permission dialog*

```
private val requestPermissionLauncher = rememberLauncherForActivityResult(  
    ActivityResultContracts.RequestPermission())
```

*// Callback for the result from requesting permission*

```
{ isGranted: Boolean ->
```

```
    if (isGranted) {
```

```
        // Permission is granted. Enable My Location button on the map  
        mapViewModel.enableMyLocation()
```

```
    }
```

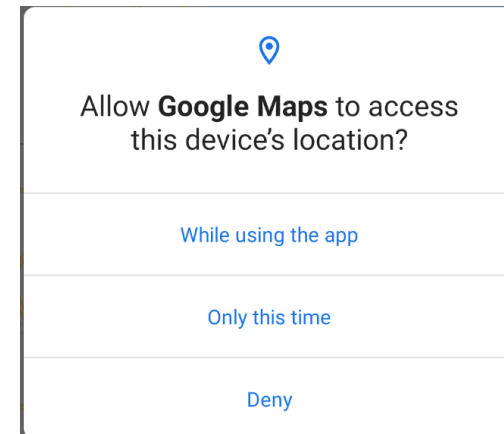
```
}
```

```
...
```

*// Ask for the permission to access the user's device location*

*// The registered call back gets the result of this request*

```
requestPermissionLauncher.launch(Manifest.permission.ACCESS_FINE_LOCATION)
```



# Geocoding

- **Geocoding** is the process of converting an address (e.g., location name or a street address) into geographic coordinates (lat, lng), which you can use to place markers on a map, or zoom to that location on the map

Hamad International Airport @ Lat:  
25.2608759 & Long: 51.613841699999995

```
/*  
    Geocoding = converting an address or location name (like a street address) into  
    geographic coordinates (lat, lng)  
*/  
private fun getGeoCoordinates(locationAddress: String): GeoLocation? {  
    val geocoder = Geocoder(this)  
    val coordinates = geocoder.getFromLocationName(locationAddress, 1)  
    return if (coordinates != null && coordinates.size > 0) {  
        val latitude = coordinates[0].latitude  
        val longitude = coordinates[0].longitude  
        GeoLocation(latitude, longitude)  
    } else {  
        null  
    }  
}
```



# Reverse Geocoding

Lat: 25.2609 & Long: 51.6138 is Hamad International Airport, Doha, Qatar

- **Reverse geocoding** is the process of converting geographic coordinates (lat, lng) into a human-readable location address

```
/*  
Reverse geocoding = converting geographic coordinates (lat, lng)  
into a human-readable location address  
*/  
fun getLocation(lat: Double, lng: Double): Location? {  
    val geocoder = Geocoder(appContext)  
    val locations = geocoder.getFromLocation(lat, lng, 1)  
  
    return if (locations != null && locations.size > 0) {  
        val name = locations[0]?.featureName ?: ""  
        val city = locations[0]?.locality ?: ""  
        val country = locations[0]?.countryName ?: ""  
        Location(name, city, country, lat, lng)  
    } else {  
        null  
    }  
}
```

# Resources

- Android Google Maps Codelab
  - <https://developers.google.com/codelabs/maps-platform/maps-platform-101-android>
  - <https://codelabs.developers.google.com/codelabs/advanced-android-kotlin-training-maps>
- Google Maps Android samples
  - <https://github.com/googlemaps/android-samples>
- Receive location updates in Android with Kotlin Codelab
  - <https://codelabs.developers.google.com/codelabs/while-in-use-location/>