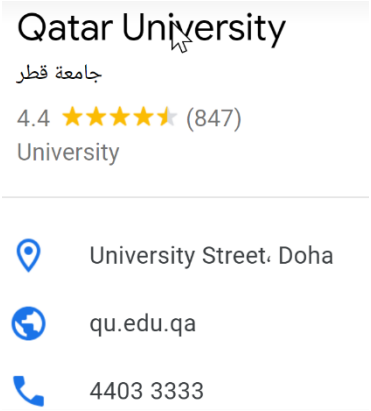

Google Maps

Google Maps Platform Key Services

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

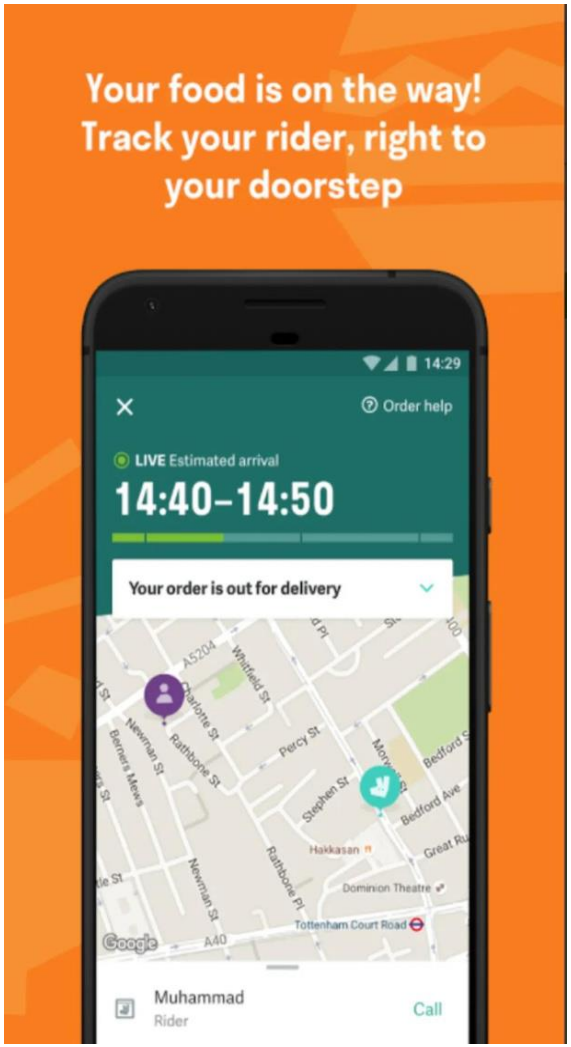
- **Routes API** :
  - Allow users to find the best route to get from A to Z using public transport, driving, biking, or walking.
  - Compute travel times and distances
  - Real-time traffic updates about the selected route

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

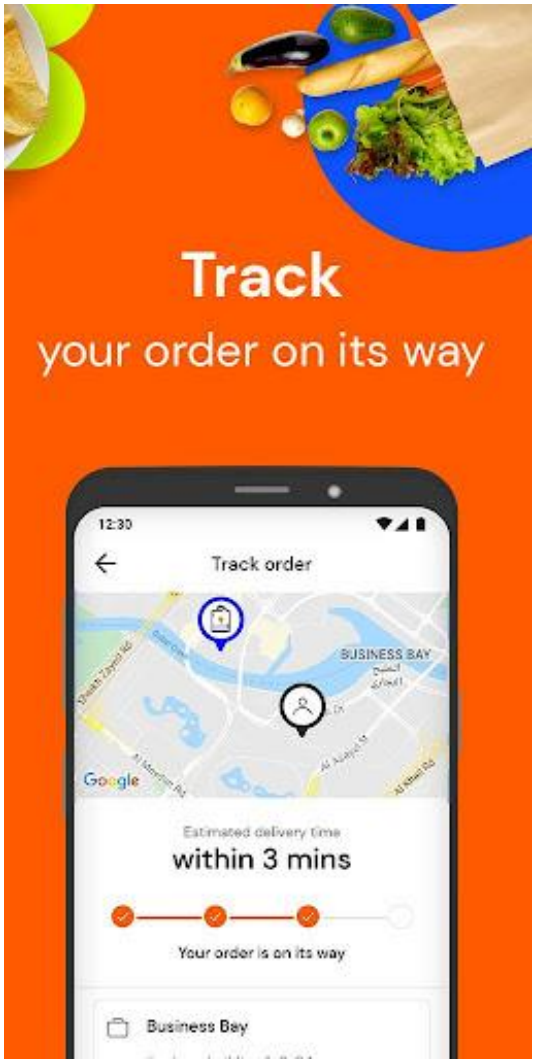


What's driving growth of Map Apps?

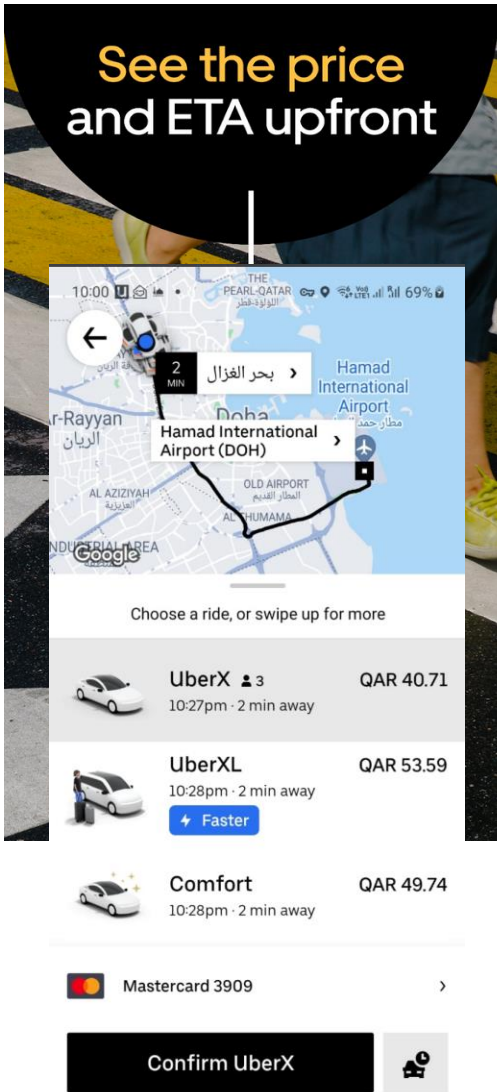
Grocery delivery apps



Food delivery apps



Ride hailing apps



Dependencies

- Add this package to **`pubspec.yaml`**

  `google_maps_flutter`

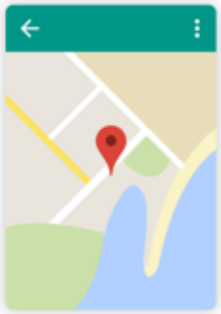
- Get Google Maps API Key. See further details at this link https://developers.google.com/maps/documentation/android-sdk/get-api-key

- Add Google Maps API key to `\android\app\src\main\`***`AndroidManifest.xml`*** file

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** widget 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

```
const LatLng quPosition = LatLng(25.377, 51.491);
const double zoomLevel = 20.0; // Buildings

GoogleMap(
   initialCameraPosition: CameraPosition(
         target: quPosition,
         zoom: zoomLevel,
      ),
   markers: {
       Marker(
         markerId: MarkerId('quMarker'),
          position: quPosition,
          infoWindow: InfoWindow(
            title: "QU",
            snippet: "Qatar University",
          ),
        ),
    },
)
```

Zoom to a Location

- **GoogleMap** widget **initialCameraPosition** 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

```dart
// Define initial position and zoom level
final LatLng quPosition = const
LatLng(25.377, 51.491);
final double zoomLevel = 20.0;

GoogleMap(
  initialCameraPosition:
    CameraPosition(
      target: quPosition,
      zoom: zoomLevel,
    ),
)
```

Add Marker

- Marker identify a location on the map at a particular geo coordinates
  - Use the **Marker** widget 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
final snippetText = "Lat: ${quLocation.latitude},
        Long: ${quLocation.longitude}"
final marker = Marker(
    markerId: MarkerId('quMarker'),
    position: quLocation,
    infoWindow: InfoWindow(
       title = "Qatar University",
       snippet = snippetText
    ),
)
```

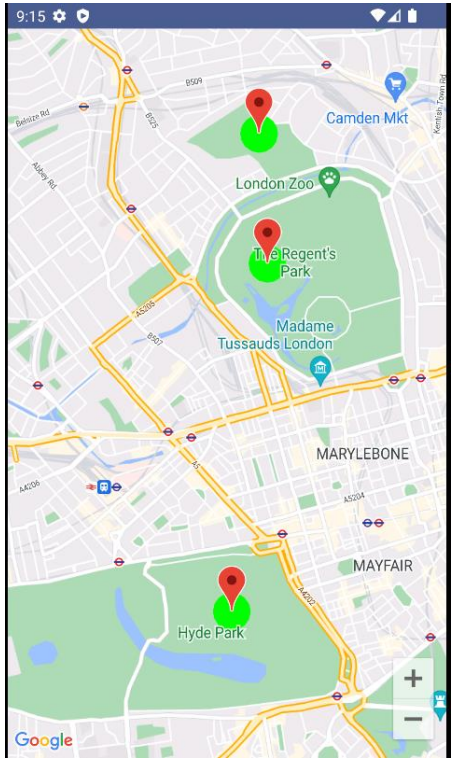
Map Customization

- Configuring the map UI
  - mapType could be `normal`, `satellite`, `terrain`, `hybrid`

```
GoogleMap(
    cameraPositionState = cameraPositionState,
    mapType: MapType.hybrid,
    myLocationEnabled: true,
    mapToolbarEnabled: true,
    zoomControlsEnabled: true,
)
```

Drawing Shapes

```
GoogleMap(...
        polygons: {
           Polygon(
             polygonId: PolygonId('quPolygon'),
             points: [
               LatLng(25.376, 51.490),
               LatLng(25.378, 51.490),
               LatLng(25.378, 51.492),
               LatLng(25.376, 51.492),
             ],
             fillColor: Colors.green.withOpacity(0.5),
             strokeWidth: 2,
             strokeColor: Colors.green,
           ),
        },
        circles: {
           Circle(
             circleId: CircleId('quCircle'),
             center: quPosition,
             radius: 500,
             fillColor: Colors.blue.withOpacity(0.5),
             strokeWidth: 2,
             strokeColor: Colors.blue,
           ),
        },
```

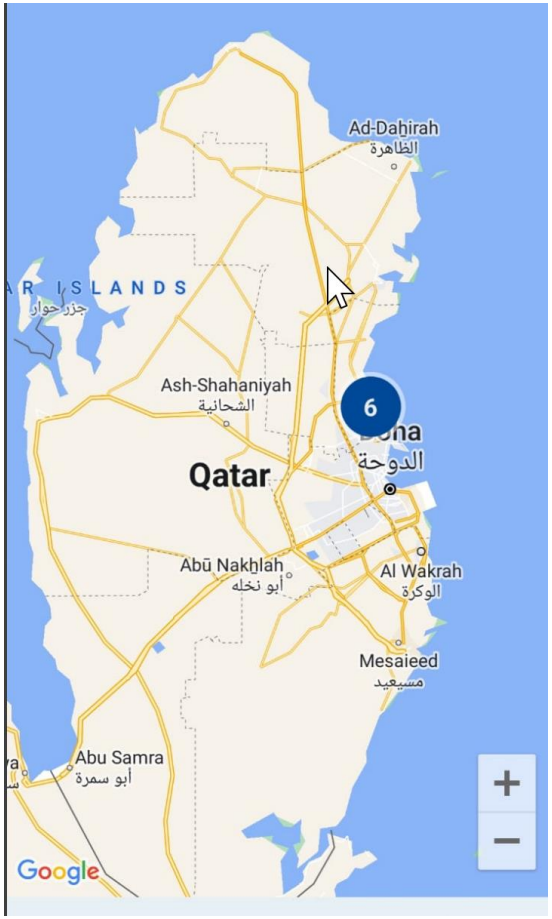


Markers Cluster

- 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.

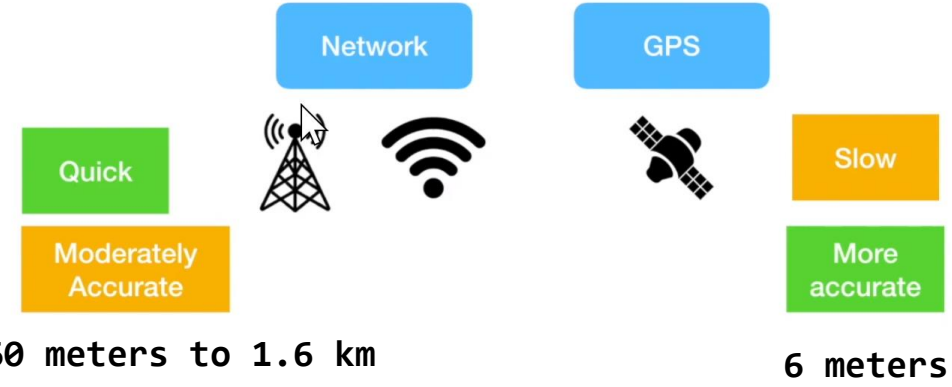
```
    val items = remember { mutableStateListOf<MapClusterItem>() }   Out Of Date  ⚠1 ✓1
    LaunchedEffect(Unit) {  this: CoroutineScope
        items.add(MapClusterItem(quPosition,  itemTitle: "Qatar University"))
        items.add(MapClusterItem(hamadAirportPosition,  itemTitle: "Hamad International 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,
```



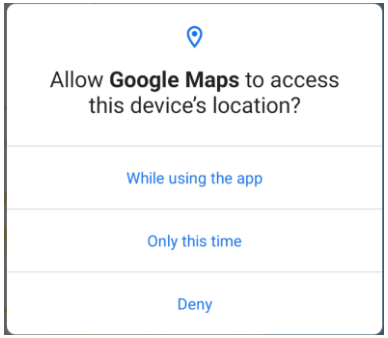
Get User Location

- Request last known location of the user's device
  - Location can determined by the **geolocator** package using WiFi & Cellular Tower and/or GPS (Global Positioning System)
  - At runtime must ask for the permission to access the device's location using




```
// Request location permission
LocationPermission permission = await Geolocator.requestPermission();
if (permission == LocationPermission.denied ||
    permission == LocationPermission.deniedForever) {
  print("Location permissions are denied.");
  return;
}

// Get the last known location
Position? position = await Geolocator.getCurrentPosition(
    desiredAccuracy: LocationAccuracy.high);
```



Geocoding

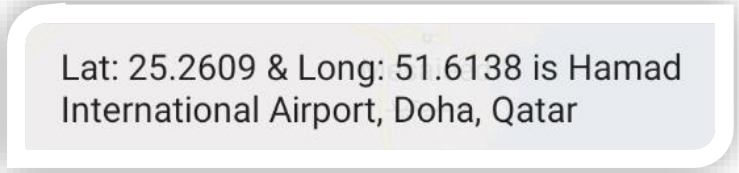
- **Geocoding** is the process of converting an address (e.g., location name or a street address) into geographic coordinates (lat, lng), which you can use to place markers on a map, or zoom to that location on the map


```
/*
    Geocoding = converting an address or location name (like a street address) into
    geographic coordinates (lat, lng) - geocoding package
*/
List<Location> locations = await
        locationFromAddress(locationAddress);

    if (locations.isNotEmpty) {
      double latitude = locations[0].latitude;
      double longitude = locations[0].longitude;

      return GeoLocation(latitude, longitude);
    } else {
      return null; // No location found for the address
    }
```

Reverse Geocoding



- **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
*/
// Get a list of places from latitude and longitude
List<Placemark> placemarks = await
        placemarkFromCoordinates(lat, lng);

    if (placemarks.isNotEmpty) {
      String name = placemarks[0].name ?? "";
      String city = placemarks[0].locality ?? "";
      String country = placemarks[0].country ?? "";
    }
```

Resources

- Adding Google Maps to a Flutter app
  - https://codelabs.developers.google.com/codelabs/google-maps-in-flutter
- A Comprehensive Guide to Using Google Maps in Flutter
  - https://medium.com/@samra.sajjad0001/a-comprehensive-guide-to-using-google-maps-in-flutter-3fbc0f7d469e