Michael Coffin

Final Project Report

iPoint

Project Statement:

The application I set out to develop was a google map app that would find your location and upon entering text in the search bar, it would implement an autocomplete feature for points of interest. I wanted to develop this app to give a more custom UI and point out points of interest within your area. I also wanted to learn more about all the google play services and see what else I could potentially do. People who travel a lot would benefit from this app to help find local restaurants, hospitals, and parks for example. I feel like my app is a good foundation to expand upon and implement other features available through the google maps API. For example, maybe the user can create a custom list of points of interest and save those coordinates for later use. So right now, there is a bunch of things to make the app better.

Application design:

The MainActivity is strictly used to handle permissions. With a onCreate() method that checks if permission granted then start MapActivity. I’m using a third-party implementation ‘com. karumi: dexter:5.0.0’ to handle permissions. Dexter is an Android library that simplifies the process of requesting permissions at runtime. Android Marshmallow includes a new functionality to let users grant or deny permissions when running an app instead of granting them all when installing it. This approach gives the user more control over applications but requires developers to add lots of code to support it.The official API is heavily coupled with the Activity class. Dexter frees your permission code from your activities and lets you write that logic anywhere you want. It makes things a bit easier, it offers some PermissionListener implementations to perform recurrent actions. BasePermissionListener to make it easier to implement only the methods you want. DialogOnDeniedPermissionListener to show a configurable dialog whenever the user rejects a permission request. MapActivity implements OnMapReadyCallback, we create a map object declaring FusedLocationProviderClient responsible for fetching current location of device. PlacesClient loads suggested P.O.I based on location. An onCreate() method for initialization of objects and buttons along with implementation of the searching part with a textChangeListener() to handle the searching part. A suggestionsClickListener() to handle the suggestions for the search bar. OnItemClickedListener() to handle clickable suggestions and moving the camera to those coordinates. A callback function onMapReady() to load the map and retrieve location of device. Also checks for gps and if it is enabled. There is a getDeviceLocation() method to handle device location. Upon fetching user location, it will move the camera to the location. The app is designed for both mobile and tablet. And the app relies on google maps api and google places api.

Application Implementation and Evaluation:

The MainActivity sets up the user to only must grant permission once and if the permission is already available go straight to map activity. For me to do this I am using a custom third-party library called dexter, version 5.0.0, which is available through github. This handles onPermissionGranted and onPermissionDenied. Once the user grants permission, app goes straight to map activity. The map activity implements google maps api and google places api, places uses the context and google api key which is a service that will return information about places using http requests. Places are defined within this API as establishments, geographic locations, or prominent points of interests. I implemented another third-party library for the search bar which also implements a autocomplete request when typing in the field. The third-party search bar is available on github and called MaterialSearchBar, version 0.8.1. When the auto suggestion comes up and once the user clicks on the place, we need location place id and send it to google api to request latitude and longitude. OnMapReady is a callback function called when the map is ready and loaded and allow us to implement what we need to do on map ready. We suppress this function because it wants to make sure it has the permission in place which are already granted in the main activity. Here we enable location with a get current users location button. This will also check if gps is enabled or not and request user to enable it. Also, it will determine if the relevant system settings are enabled on the device to carry out the desired location request and invoke a dialog that allows the user to enable the necessary location settings with a single tap. Then checks whether current location settings are satisfied. Theres two possible outcomes we need to handle is success and failure to retrieve device location. On success, call getDeviceLocation which this again we need to suppress because it depends on permission which is handled in main activity.

Experiences and Thoughts:

So, the reason why I chose to do a google maps app is because I thought that this was the most interesting subject taught in the class. I wanted to further expand my knowledge on the google play services api for future projects that I have in mind that would be practical in the real world. So committing the time to this really helped build a foundation not only on future projects but on this app. The third party api’s really helped simplify things during the process. I wanted to further expand on the points of interest concept with this app such as directions and saving the list and storing and using that data through a server and do this on a separate activity. I did make the attempt to do this with a P.O.I button but could not get it up and running due to time constraint.