Implementing the LitterBug App

Litter has a negative impact on the environment and contributes heavily to water and soil pollution. As litter degrades, it releases unnatural chemicals and microparticles into the environment. While awareness about litter and the environment has become more prevalent and it seems that more people care about the impact humans are having on the environment, I've noticed that the amount of litter in my community hasn't changed and that people are still unwilling to pick up litter. I spend a lot of time going on walks in my town and neighboring towns. I've noticed that no matter where I'm walking, in a neighborhood, on a busy street, and even in the woods, there is always litter that needs to be picked up.

A lot of people care about the environment and the health and well being of their communities, but are unmotivated to or have never picked up litter. I wanted to create an app that had a positive impact on the environment and that motivated people to help the environment through trash pickup.The LitterBug app is for people who want to make their communities more beautiful and safe. My goal was to create an app that allows users to be able to track their litter pickup on a map and see their impact. I also wanted to facilitate the users' cleanup by allowing them to see litter pinned by other users that needs to be picked up. This feature would also allow people that can't pick up litter to still have a positive impact on the environment and to help others more easily find litter that needs to be picked up.

I began my app creation process by designing three tabs for interfacing with the user: the Home tab, the Impact tab and the Litter Map tab. After completing the design layout of the tabs I created the bottom navigation bar that includes the three tabs. The home tab is where the user can see a summary of the amount of litter they've picked up. The user can view a map with the pinned locations of all the litter they've picked up on the Impact tab. The Litter Map tab displays a map with pins showing locations where litter needs to be picked up. On this tab the user can push a button to indicate that litter has been found at their current location; they can also select a pin and indicate that they have picked up litter. Once the litter is picked up it is added to the user's Impact map.

The Impact and Litter Map tabs are the most important and crucial components of my app. To create these so that users can map litter and see their mapped litter I used the Maps SDK for Android. To begin creating my maps I first had to obtain a Google Maps API key from the Google API console. Once I obtained my API key I could then create my map. To do this I used my Bottom navigation fragments that I created to act as containers for my two maps. I then got a handle to the map fragments by calling FragmentManager.findFragmentById(). Then I used getMapAsync() to register for the map callbacks. Next I Implemented the OnMapReadyCallback interface and overrode the onMapReady() method, to set up the map when the GoogleMap object is available.

The next step in the creation of my app is to get the user's current location when they want to pin a piece of litter so that it can automatically pin the litter without the user having to input their location themselves. To get the users current location I first needed to request permission to use

their location in order to preserve privacy. To do this I used RequestPermission, included with AndroidX , that allows the system to manage the permission request code for me. When user wants to pin a location where litter has been found they will press a floating action button located on the Google Map fragment. When the user presses the button to tag the litter location, LitterBug will store the user's current location as long as the user gives location permission. I used FusedLocationProviderClient and LocationRequest Java classes to get the user's last known location.

In order to maintain user data, the litter location data and the user's litter pickup history I needed to incorporate some kind of backend database functionality. To do this I used FireBase as a Mobile Backend as a Service (MBaaS). A Mobile Backend as a Service is "a cloud computing platform that provides predesigned and ready-to-use backend functionality via a customizable software development kit or application programming interface". MBaaS platforms typically include the framework and components usually found in a typical mobile server, like data storage, geolocation, authentication, push notifications, and user management. I originally didn't know how I was going to store my litter data or my users data and I found that there were two options, I could use a MBaaS or I could develop my own mobile backend from scratch. The latter method is much more difficult and more time-consuming so I decided to use a Mobile Backend as a Service in order to simplify server-side development. I decided to use Firebase for my MBaas and used a Firebase Firestore Database to manage the data for the Litterbug app.

I was looking at other Mobile Backends as a service, but eventually decided on Firebase. AWS Amplify has similar features to Firebase, but firebase seemed like it was beginner friendly and a lot easier to use than AWS Amplify. I also considered using Backendless. Backendless uses SQL where Firebase and AWS Amplify do not. Backendless, like AWS Amplify, seemed like it would be more difficult to use than Firebase. Firebase is a Google product and already built into Android Studio. The only downside I've found to Firebase is that it isn't suitable for large scale applications with a lot of data, but my project won't be holding that much data.

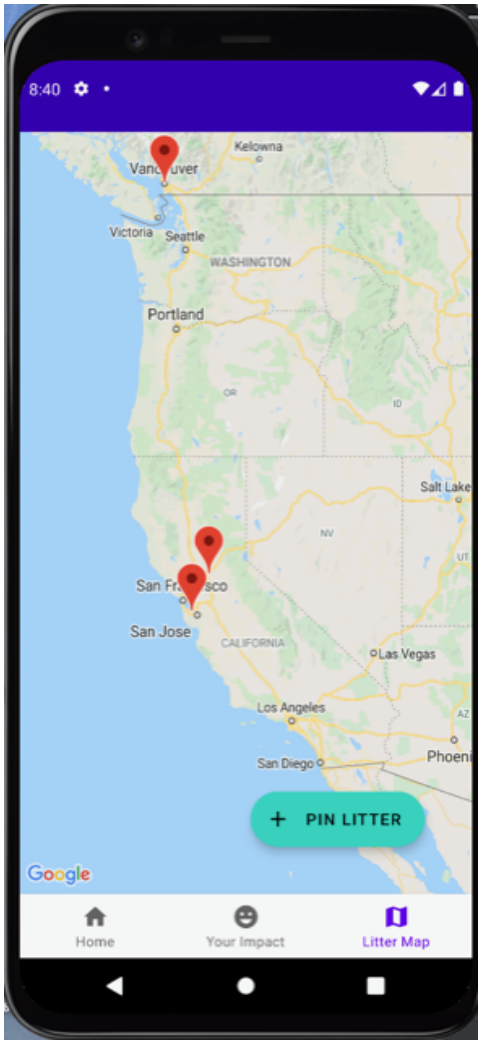
I decided to use Firebase Firstore Database for my app because it is an easy to use Mobile Backend as a Service. Firebase Firestore Database stores and syncs data with a NoSQL cloud database. Firebase Firestore Database allows my litter data to be synced across all of my users in realtime, and remains available when my app goes offline. Firebase Realtime Database is also provided by Firebase. I chose to use Firestore Database over Firetore's Realtime Database because it is more structured than Realtime Database. Firestore Database contains collections of data which include documents. Within each document are data fields. For my LitterBug app I included a collection called litterLocations for my Litter Map tab which holds documents for each of the locations where litter is located. Each document contains the latitude and longitude of the litter location. Similarly, there is a pickupLocations collection for each Litterbug user that stores the latitude and longitude for each location where the user picked up litter.

After deciding I was going to use a FireBase Firestore DataBase to hold the litter location of each of my users cleanups I needed to begin implementation. I found that I needed a way to

index into my Firestore Database by location. I couldn't index into my database when I stored the users location as separate latitude and longitude fields. I needed a way to store latitude and longitude into one single field and be able to index to that field and get the location. I found that I could do this using Geohash included with the GeoFire library. GeoFire allows you to store and query a set of keys based on their geographic location and Geohash is a system for encoding a (latitude, longitude) pair into a single Base32 string. I was able to create and encode a string based on the latitude and longitude by using GeoFireUtils.getGeoHashForLocation(new GeoLocation(lat,lng)). The Geohash string is used as the means to index into both the litterLocations and pickupLocations collections . When a user picks up a piece of litter in the Litter Map navigation tab I have to remove that pinned litter location from my litterLocations Firestore Database collection and add it to my user's pickupLocations collection. Geohash provides the key for updating the collections within the database.

The next step in my app creation process was to create a sign in page. To do this I first created my own sign in user interface, but soon realized I was unsure of how I was going to store my users data. With some research I found that firebase allows you to authenticate users with secure logins that track and identify every user who logs in and out of my app. I also found that FirebaseUI handles both the authentication and user interface of the applications sign in page. I decided that using FireBaseUI was the best sign in option for my app. FirebaseUI authentication allows my LitterBug app to authenticate users without me needing to implement my own backend. FireBaseUI requires all API requests to communicate with Firebase Authentication services. FirebaseUI also handles all the user interfaces for each of the different service providers. I only used email and password authentication, but FireBaseUI provides google, twitter, facebook, and phone authentication options. FireBaseUI also handles user account creation and connection in addition to the linking of accounts from different providers automatically.

There were many features that I would have liked to include in my app but was not able to because of time constraints. I would have liked to have allowed the user to include data about the litter they picked up so that the data could have possibly been used to help communities and policy makers create changes accordingly. I would have liked the user to be able to include an image of the litter as well as a description of the litter and whether the litter was potentially hazardous. I would have also liked to have pinned nearby garbage disposal sites so that users picking up litter could easily find a place to throw away what they picked up. I also wanted to include a navigation tab in my app that allowed users to join litter pickup challenges with friends, family, co-workers, and other community members. I thought that this feature would have helped motivate users to keep picking up litter.I also wanted to create notifications for when there was new litter pinned in a location near a user. Lastly, I would have added another navigation tab on my app where users could message one another to schedule and plan meetups for picking up litter in their communities.



Sources:

https://www.quora.com/How-do-I-develop-the-server-side-of-an-Android-application

https://www.mobindustry.net/blog/top-9-mobile-backend-as-a-service-mbaas-platforms/

https://hitherejoe.medium.com/exploring-firebase-ui-on-android-authentication-7a6fc6b7e35