Introduction:

As the world is seeing increases in both the traffic congestion on the city streets, and the impacts of global warming - all while gas prices are on the rise once again, bike commuting has a lot to offer. The problem is, there are not many prominent bicycle parallels to car service centers, autopart stores, and roadside assistance services. So in the case of a novice cyclist contemplating the transition to commuting by bike, one may be deterred by the worry of a mechanical issue causing a delay on their ride to work. Even in the case of an experienced biker who is confident in their repair skills, the practicality and convenience of carrying a toolbox on a bike at all times is minimal. BikeAlert is an android app that is focused on connecting the biking community with a suite of available resources to assist both novice and experienced riders, should they encounter any setbacks while in transit. Many times the issues one is faced with on a bike ride require relatively simple fixes, requiring negligible time to repair given the proper tools or training. Therefore this software development team has designed BikeAlert to make it easier for bikers to find the means necessary to get the wheels rolling again. With features such as the ability to add details of each of your bikes to your profile, locate a nearby bike shop or repair station, find other riders and add them as friends, set issue markers on the map for your friends when something goes wrong or go private and take yourself off the map when you want to ride incognito - BikeAlert has all the essentials you need in your mobile bike toolbox.

Research:

The ideology of this app was primarily focused around sourcing the solution to a community-wide problem from members of the community as opposed to relying on specialists from the outside who would offer guaranteed solutions at a high price. There have been many applications in the last few years that have aimed to do this in various industries and all typically revolve around more efficiently allocating resources by presenting consumers an option to go with a less legitimate service from someone who is not a specialist but who can do an adequate job at a considerably lower price.

The most successful example of sort of resource comes from the rideshare companies Uber and Lyft. Driving to a location is not something that most people consider to be difficult but for lots of people, especially people in large cities, the issue comes from not having access to a car. Before smartphones and assuming public transportation is not available, an individual's best option is to call a taxi with the ability to charge an extremely high premium for the convenience of this service. While in this case there is the tricky issue of taxi medallions helping justify these premiums, the main reason taxis (historically) have been such a popular service is because though many individuals are willing and able to drive people to their destination it was extremely difficult to match these people with those who were in need of a ride quickly enough. Though it is unfortunate for the taxi drivers, once this issue of allocation was solved and Uber/Lyft were able to match up drivers with people who want to be driven, customers were no longer willing to pay the extra money for a taxi because even though they were the *professionals* they didn't do a better enough job than any random person at driving from point a to point b.

Due to the nature of the service, Uber and Lyft are able to allow essentially anyone who has an acceptable vehicle to drive for them and make money doing a task they probably already do most days. Our app is seeking to do a similar thing by trying to empower members of the cycling community to both network with each other but more specifically to learn things and help one another with quick, simple repairs. Many times issues that arise while commuting by bicycle are extremely quick to fix so more often than not the issue is a lack of either the knowledge to diagnose the problem and attempt a fix or the tools to address the problem correctly. While this type of allocation issue is extremely similar to Uber and Lyft, the service we are seeking to help provide is closer to the one provided by the vehicle services company AAA. AAA differs from rideshare companies in that the people who work for them are typically not contractors but rather full time employees that are considered professionals by most customers. The tasks that are expected of a AAA employee are typically beyond the capabilities of the customer, whether that is because they lack the knowledge or tools to change a tire or because their vehicle is in need of a repair that cannot be done easily on the side of the road and needs to be towed to a mechanic.

Regardless for both types of companies, it makes the most sense to formally split the users into two categories based off how they intend to use the app, one category for users and one category for service professionals. We decided to not go that direction for a number of reasons but the primary one was that we felt it was unnecessary because the service provided would not involve any kind of quid pro quo and therefore there would be no liability assumed by the either end user or by the app. While this might seem seem unreasonable to rely purely on the altruism of the community there is actually already a popular service in the cycling community based off a similar model called Warmshowers which is an app designed for people to open their house up and host cyclists touring across the country at their home temporarily. Warmshowers is made for people who are into bicycle touring to participate and share in the interesting experiences of other folks in the same community and it works extremely well because at some point most people who use the service will both be a host and be hosted. We feel the same way about our app in that even the most prepared bicycle enthusiast will have a mechanical failure at some point when they aren't prepared and will need help and ever they have a positive experience will be more than happy to pay it forward. So a combination of all of these apps or services is what we are going for. We would like to solve an allocation problem by providing a tool that allows the community to call upon reserves of specialized labor that are already willing and able to provide a service often times completely altruistically but other times with the hope of future reciprocation if that need were to arise.

Development Timeline:

This project idea was suggested by Michael. Being an avid bike rider, he figured there was a need to create an app that could allow him to help his own fellow bike riders. The group got together and had a conversation regarding thoughts on the idea. Everyone seemed to be happy with the pitch, so we started the requirements and analysis phase of development. At this point, we had a basic coding plan laid out and each member chose a piece of the target to work on. The project's development began by implementing the Google Maps API, considering it is the focal point of the entire app. Reading through the literature of the Google Maps API, in conjunction with Zach Yannes' lectures, the team was able to implement the basic inner working of the MapActivity. Following this portion of coding, the group encountered the first instance where the specifications needing to be modified based on what we now knew to be realistic (or unrealistic) goals for the target product. The final portion of this phase was to instantiate basic (be that, essential) map features like zoom, and adding a Hamburger Menu (Navigation Drawer).

Now that a rapid-prototype for the app was established, the team focused efforts on connecting what was currently only a local app with Google Authentication Services and Firebase Cloud Firestore. It took several days for both of these to be in full working order. Once the authentication and Firebase database were established, the next question entailed the strategy by which the database would be structured. The group's consensus was to have a collection of Riders with document ID's being their emails. Inside of these emails we would store the information for their Bike, location, pendingInvites, and friendList.

With the back-end now set up, it was time to refocus efforts on front-end development. At this time, each member of the team diverged and began working on individually-assigned tasks. This lead to the formation of the following activities: addBikeInfo, settings, and user. These instantiations were just basic activities written to either display data or send data to and from the Firebase database. After these basic classes were constructed and implemented, focus was given to develop what would be the core purpose of the app: "Request Help".

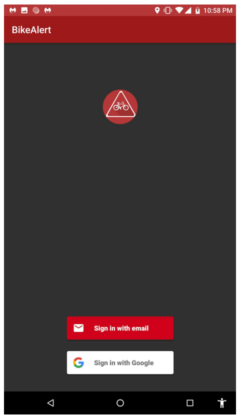
At this point the group needed to determine how to add a search bar and markers to the map. After some trial and error, the group decided on a method - which we believed - could get these processes function properly. In order for this to happen, creation of a new collection in our database for requests for help was required. Furthermore, code was written to store and retrieve a user's location from the Firebase database. Not only did this allow the group to create help markers, it also enabled the tracking of all riders in real time.

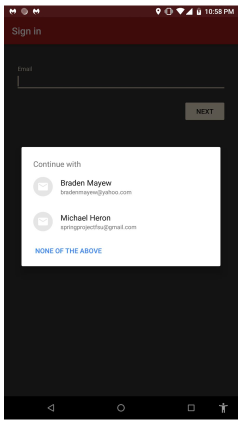
At this phase of development, there were not many more features to add, apart from a friends list, pending friends list, and cleaning up the user interface. Formation of the friends list took much longer than expected. Several hurdles needed to be overcome, including not knowing how to structure the database to reflect and handle "friends" (in general), as well as having issues querying the necessary information once the pending friends and friends lists were created. Days were spent working with various types of adapters, recycler views, and list views, oftentimes just to scrap the code in the days that followed. At this point, Zach Yannes was approached by members of the team to see if he had any insight as to how the group should approach instantiating a working friends list. He managed to point the team in a different direction. Following that meeting, it was determined that the Firestore Recycler View would be used in conjunction with some other custom Recycler Views and Adapters. At this point, BikeAlert has a fully operational friends list and pending invites list. The last day was spent performing non-execution based testing, i.e., code walkthroughs were conducting in an effort to isolate any "code smells" that may have been overlooked. Ultimately, around fifty hours were spent developing the project: Five hours for the google maps; Twenty hours for Firebase related programming; Fifteen hours for the friends list implementation; Ten hours for UI development and Logo creation in Adobe Photoshop and Illustrator.

<u>Feature Overview:</u>

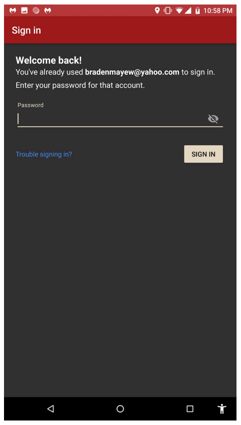
BikeAlert has a number of features that the development team considers useful in helping people get their mechanical issues fixed. The core feature of BikeAlert is to empower riders by providing them with the means to request help for a specific mechanical issue, without having to lug a bike into a shop and have it looked at. For users, this is as simple as pressing a button on the screen to create a dialogue box which allows them to enter the exact nature of their problem and display it to anyone who may offer assistance. This was implemented using a button laid over the map with an onclicklistener on the button that brought up the dialog box. Once the dialog box was closed the users information was then used to post a request into the firebase collection for requests so that it would appear on the map.

When users open the app, they are presented with the option to log in using a Google account or signing up with their email address. If the user selects sign in with email and they've used the app before, they're prompted to enter their password to sign back in; otherwise they are prompted to enter their name and a password to create a new account. From there, a new user will have to add the details of their first bike. New user and bike information is added to the Firebase database.

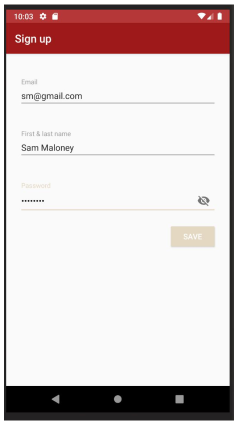




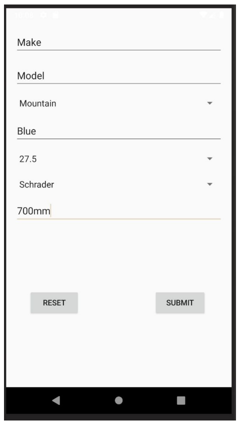
Braden Mayew
bradenmayew@yahoo.com
Michael Heron
springprojectfsu@gmail.com



Welcome back!
You've already used **bradenmayew@yahoo.com** to sign in.
Enter your password for that account.

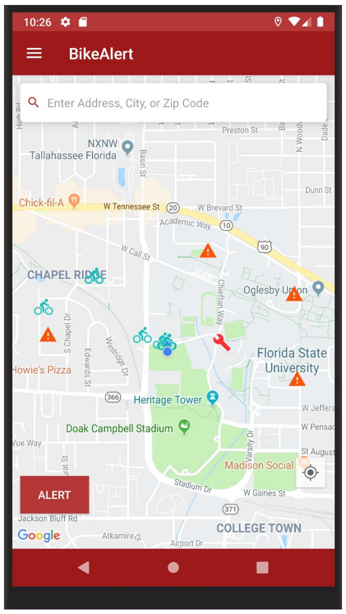


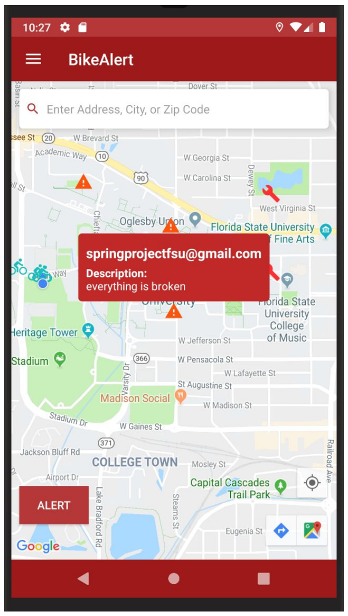
Email: sm@gmail.com
First & last name: Sam Maloney
Password: ••••••••



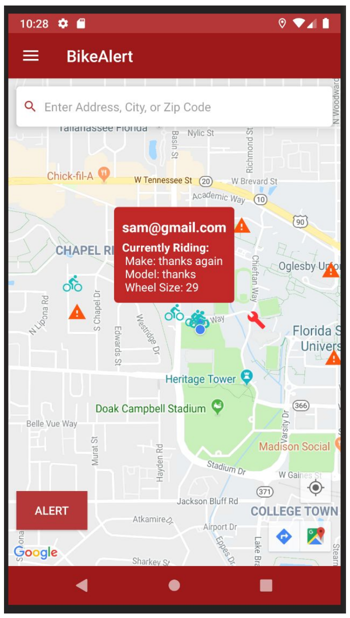
Make
Model
Mountain
Blue
27.5
Schrader
700mm

Once the user has logged into their account, they are brought to the MapActivity. On the map activity, user's current location is displayed with a blue bike on the screen. You can tap on the bikes of other users to see their email and the details of their bike. In addition to bikes, you can see the orange deltas signifying a request for help has been made from that location. You can see a description of the issue the user is having if you tap on it. There are also red wrenches on the map signifying repair stands and bike shops.

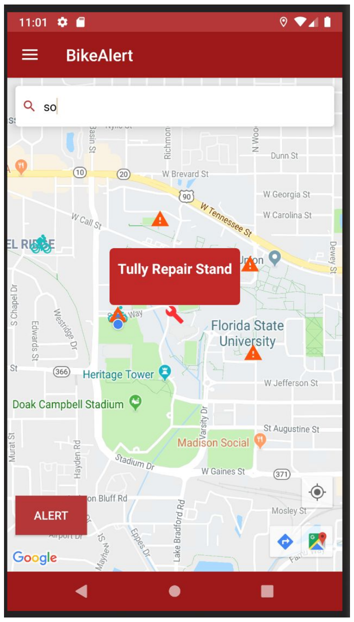




springprojectfsu@gmail.com
Description:
everything is broken

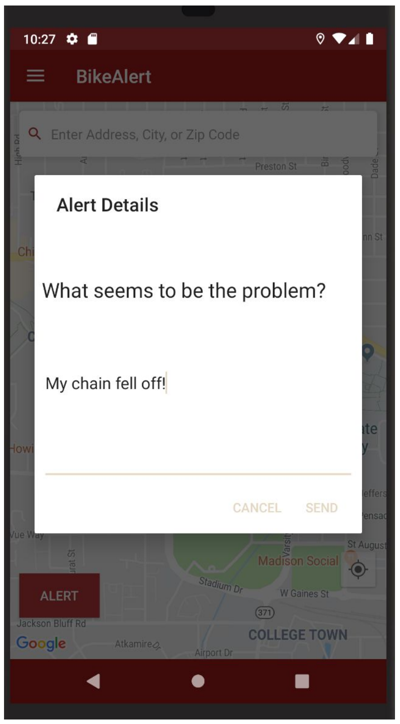


sam@gmail.com
Currently Riding:
Make: thanks again
Model: thanks
Wheel Size: 29



Tully Repair Stand

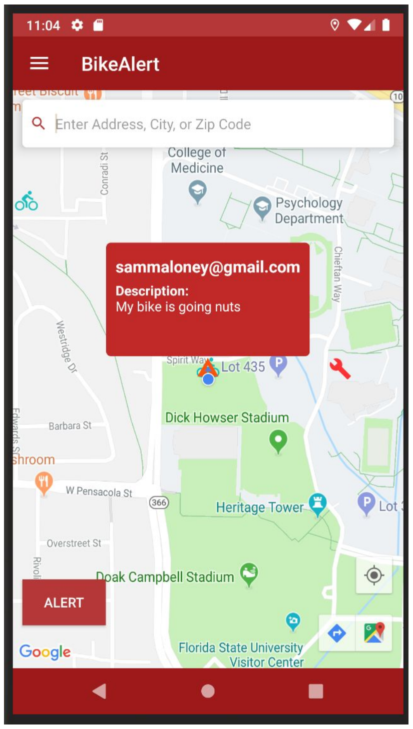
Should the user find themself in need of assistance while on a ride, you can hit the red ALERT button in the lower left-hand corner of the screen. A request dialogue appears, where the user can enter the issue they're having. Once that is complete, they are brought back the map, and a new alert (orange delta) has been added at the user's current location.



Alert Details

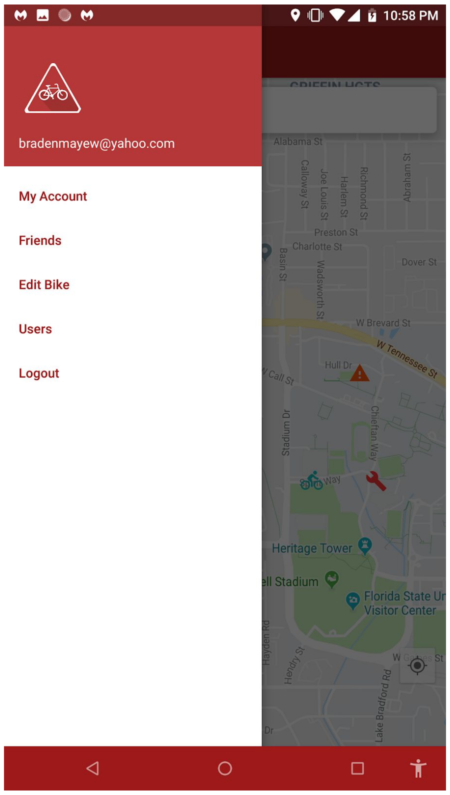
What seems to be the problem?

My chain fell off!

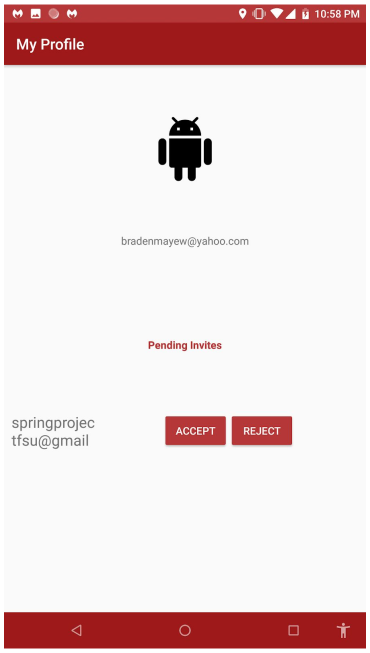


sammaloney@gmail.com
Description:
My bike is going nuts

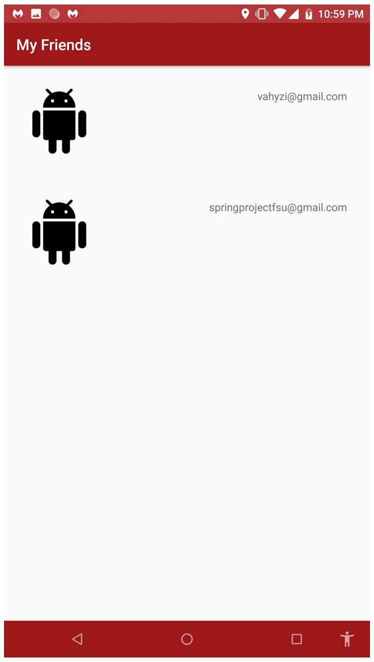
For the creation of the friends list component, it was thought necessary to incorporate three new components into the Navigation Menu: My Account, Friends, and User. So if the user swipes from left to right or selects the button in the upper left-hand corner of the screen, they are presented with the hamburger menu containing those options, as well as Edit Bike and Logout.

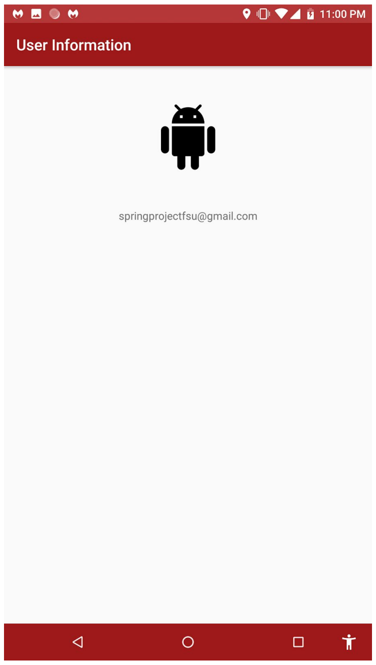


When My Account is selected in the Navigation Menu, the current user's information is displayed along with all pending friend invitations. From there the user can choose to either accept or reject the pending invites. If the friend invite is accepted, that user that requested to be friends is deleted and moved from the pendingFriends list in the Firebase database to the friendsList. If said invite is rejected, the user is deleted from the pendingInvites Recycler View as well as from the pendingInvites collection in Firebase.

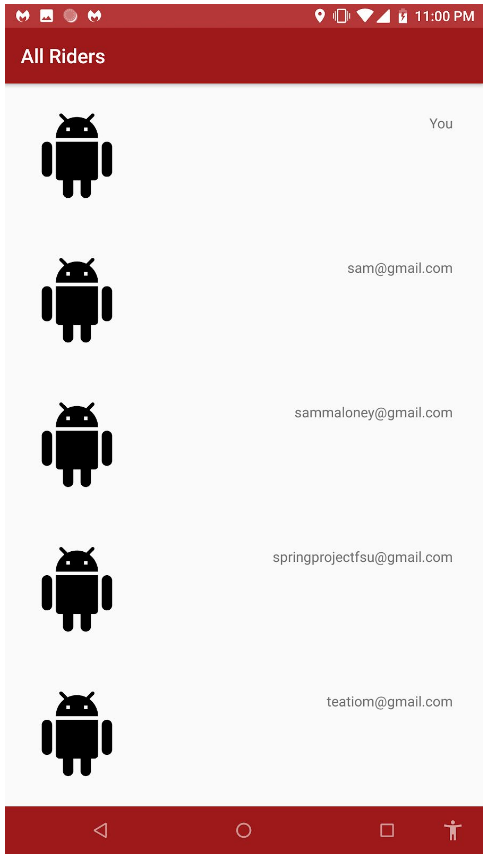


At this point, when the current user selects the Friends option contained within the Navigation Drawer it correctly displays all current and newly added friends. Furthermore, the “SendFriendRequest” button will no longer be visible for any users that are already friends with another selected user.





Finally, the Users feature of the Navigation Menu displays all users and the current user is defined as “you” and does not feature any onClick functionality. The other users do have an onClick feature that allows their information to be viewed, as well as having an option to add that user as a friend if they are not already one.



Future Development:

Due to the nature of our project, there is inherently an nearly endless number of ways we could extend the functionality of our app to potentially enrich user experience. One of the main things the want to do is really figure out a way to implement the ability to go into a “private mode” while still using our app. The issue we ran into is that we don’t think that users would be as likely to request help if they didn’t see someone in the immediate area and we also think that most people would want to not appear on the map given the option to go private. We were really unable to find an appropriate balance for this issue but some of the ideas we discussed were, only allowing users to go private for so long and so often, keeping most user data about folks on the map hidden, and only showing the number of users in your area as opposed to the exact location of the users. While these are all fine solutions that would work to varying degrees no solution would make every person on the privacy vs convenience spectrum happy and we do feel like the issue is more complicated than we initially thought and would probably need to weigh community input as our app got more users.

Now that we have a friends list implemented for our users, we would like to integrate the ability to send notifications to a user’s friends when they submit an issue. We intended to use Firebase Cloud Messaging (FCM) as the means to accomplish this task, but it has proved to be more challenging than initially intended. Due to the fact that Firebase Cloud Messaging is a fairly new replacement for Google Cloud Messaging (which has just been deprecated as of April 2019), there is a lot of conflicting information in various sources of documentation. Once going through the process of getting up FCM in the project without errors, and adding the appropriate classes and functions. Sending text messages from the Firebase Console to the emulator’s ID token was unsuccessful. Furthermore, we found that many of the methods used in the sources we referenced are no longer available in Firebase Cloud Messaging, and their updated versions are not documented. One particular source mentioned the use of Xamarin in conjunction with FCM, but the steps taken in the installation instructions were unavailable in Android studio. While notifications will be a useful feature for out users, we may need to explore other methods of implementing this feature.

We would also like to add a view quality of life changes to our friends list functionality. As of now there isn’t a way to remove friends in-app and finding new friends isn’t as easy as we would like. This is mostly because of difficulties we had updating firebase both in real-time and on user input. Lots of aspects of this app seemed like they should be easy to implement with firebase but in practice we weren’t able to get queries to consistently return the values we wanted in data formats that were easy to handle. Decisions about what should be kept local on the device as opposed to what should be sent and pulled from the database also complicated things and given time as well as seeing how these things work with more testing would better allow us to make informed choices that could be more efficient.

The final few issue with the friends implementation pertain to the user not having the ability to see the friends list of another rider. Both UserInformation screens (current and general) do not have any details apart from email. Ideally, seeing all information including their different bikes and being able to add profile pictures would be a strong enhancement.

This brings us to our last main area that we would like to add more features which is social media integration. We feel that our app is the perfect use case for allowing user to post directly onto their other social media profiles in order to increase the visibility of their issue and also to promote more people to use our app. While we feel that our app will be more helpful than just a social media post with the inclusion of the real-time map, we ultimately just want our user to be able to get their problem solved and we recognize that conventional social media apps have tremendous quantities of users which could lead to a problem getting solved faster. Being able to send out your requests to friends lists on other social media platforms might help with this and would certainly help increasing the use of our app which in turn could increase the speed and likelihood of a problem getting solved. Another easy way to integrate social media into our app would be to allow users to send invites to their BikeAlert friends list through another social media site. Again we feel that more users will make our app work more effectively and making it more convenient for users to use our features will certainly aid in that.

Conclusion:

For the majority of people to feel confident enough to get on board with bike commuting, they have to acquire a belief that biking will be just as easy and reliable as getting in a car and turning the key. For the average person, the list of what could go wrong when riding a bike is enough to convince them not to. Via an intuitive method of receiving bike assistance in an emergency, BikeAlert offers a means to reduce concern and ease the minds of those that may be on the fence about switching to bike commuting. While BikeAlert by no means reinvents the wheel, it goes a long way in offering the piece of mind desired by most people to give them the necessary push to try a new mode of transportation.

Citations:

http://ww2.cs.fsu.edu/~yannes/

https://firebase.google.com/docs/android/setup

https://codelabs.developers.google.com/codelabs/realtime-asset-tracking/index.html?index=../../index#5

https://codinginflow.com/tutorials/android/navigation-drawer/part-3-fragments

https://developers.google.com/android/reference/com/google/firebase/firestore/GeoPoint

https://firebase.google.com/docs/firestore/query-data/get-data

https://firebase.google.com/docs/firestore/query-data/listen

https://firebase.google.com/docs/reference/android/com/google/firebase/firestore/Query

https://developer.android.com/studio/debug/am-video

https://stackoverflow.com/questions/40366717/firebase-for-android-how-can-i-loop-through-a-child-for-each-chi
ld-x-do-y

https://stackoverflow.com/questions/45207709/how-to-add-marker-on-google-maps-android

https://developers.google.com/maps/documentation/android-sdk/map-with-marker

https://developers.google.com/maps/documentation/android-sdk/marker

https://developer.android.com/guide/topics/ui/dialogs

https://firebase.google.com/docs/cloud-messaging/android/client?authuser=0

https://stackoverflow.com/questions/10111073/how-to-get-a-bitmap-from-a-drawable-defined-in-a-xml/1011120
3#10111203

https://developers.google.com/maps/documentation/android-sdk/start

https://developers.google.com/maps/documentation/android-sdk/map-with-marker

https://www.androidauthority.com/android-push-notifications-with-firebase-cloud-messaging-925075/

https://www.simplifiedcoding.net/firebase-cloud-messaging-tutorial-android/