Udacity Capstone P7

Problem:

Several years back now I took an Uber and started chatting with the driver. I asked how he liked the job and he said it was overall good; however sometimes things can get chaotic. I asked if he wouldn't mind elaborating and was told of a situation where his fare would be at a large event. Arriving to a crowd of participants looking for their Uber/Lyft, it was confusing to match fare with ride. Instantaneously I solved this problem.

Solution:

I saw a scene with driver and rider phones flashing the same color and number. This was a link which could provide both parties the means to verify each other. The rider will orient their displays so that the screen is facing out towards the rides streaming by after the "your driver is arriving" is displayed. From the drivers POV, they see a card with the same pattern within their app and can view the match outside amongst the crowd. Two way verification can be had by driver tapping card to bring it full screen, rolling down the window, and showing it back to the rider.

Architectural Design Choices:

This has many more applications than the above scenario. It appears best to factor the feature set so as to compartmentalize the flashing views and their associated classes from the rest of the demo app showing off the functionality. For that reason, the base functionality will be contained within an Android library. The app project will contain all activities and code necessary to demo said function.

This app would be less effective for smaller devices. As well older devices can't make use of the design library and advanced animations. Seeing as both the average screen size and manufacturer set Android SDK level has increased as time carries on, the minimum SDK will be set at API level 19 with a target of API 21.

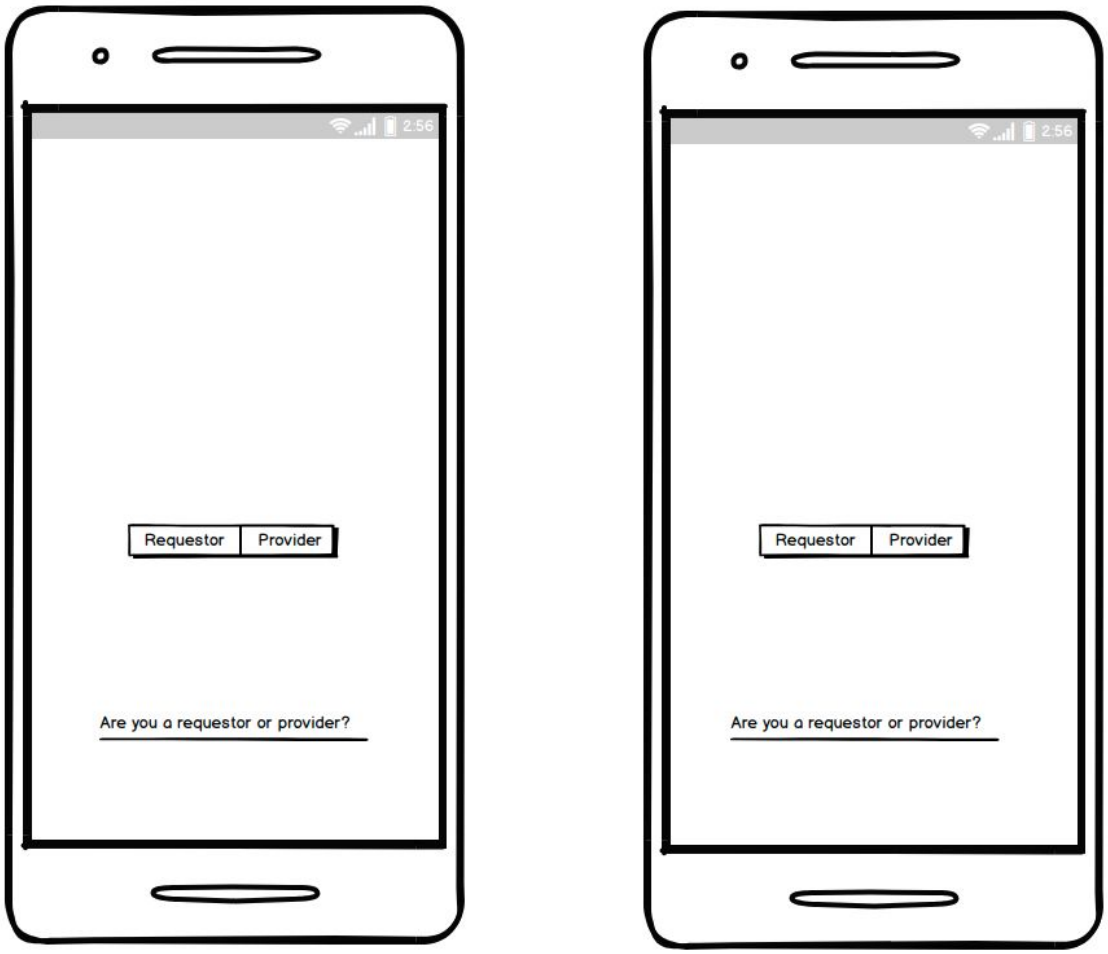
Intended Use Cases:

The above scenario is one of several. It's a good fit though several others exist. The following lists other applications (starting with the aforementioned) though they are not all. We are only limited by our imagination.

- ★ Taxiesque services
- ★ Meetup organizations
- ★ Delivery services

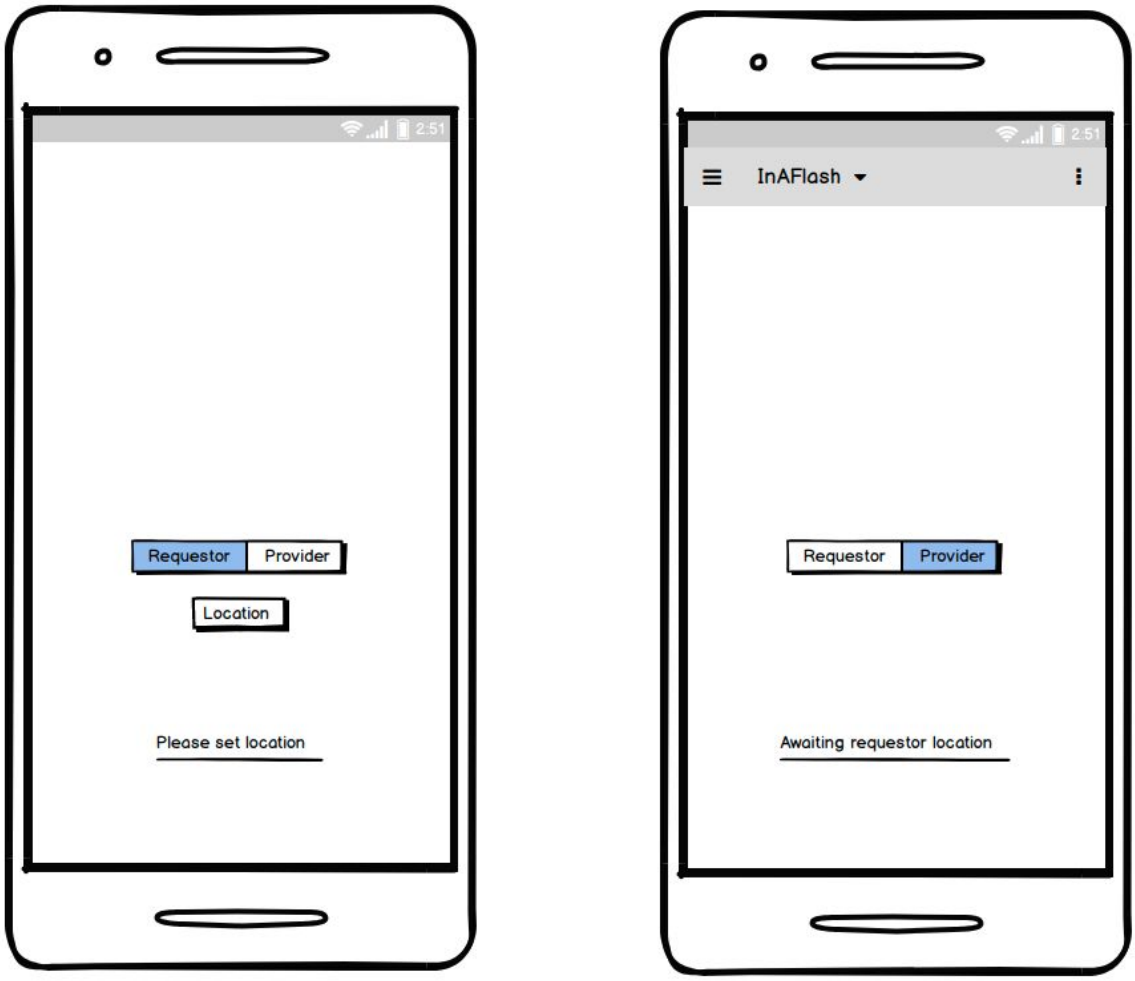
Primary Features:

- ★ Color/Number displays full screen and as card
- ★ Randomization and synchronized pair between devices
- ★ Stationary requestor/ Mobile provider model
- ★ Android Library for portability
- ★ Demonstration app showing a common scenario for usage



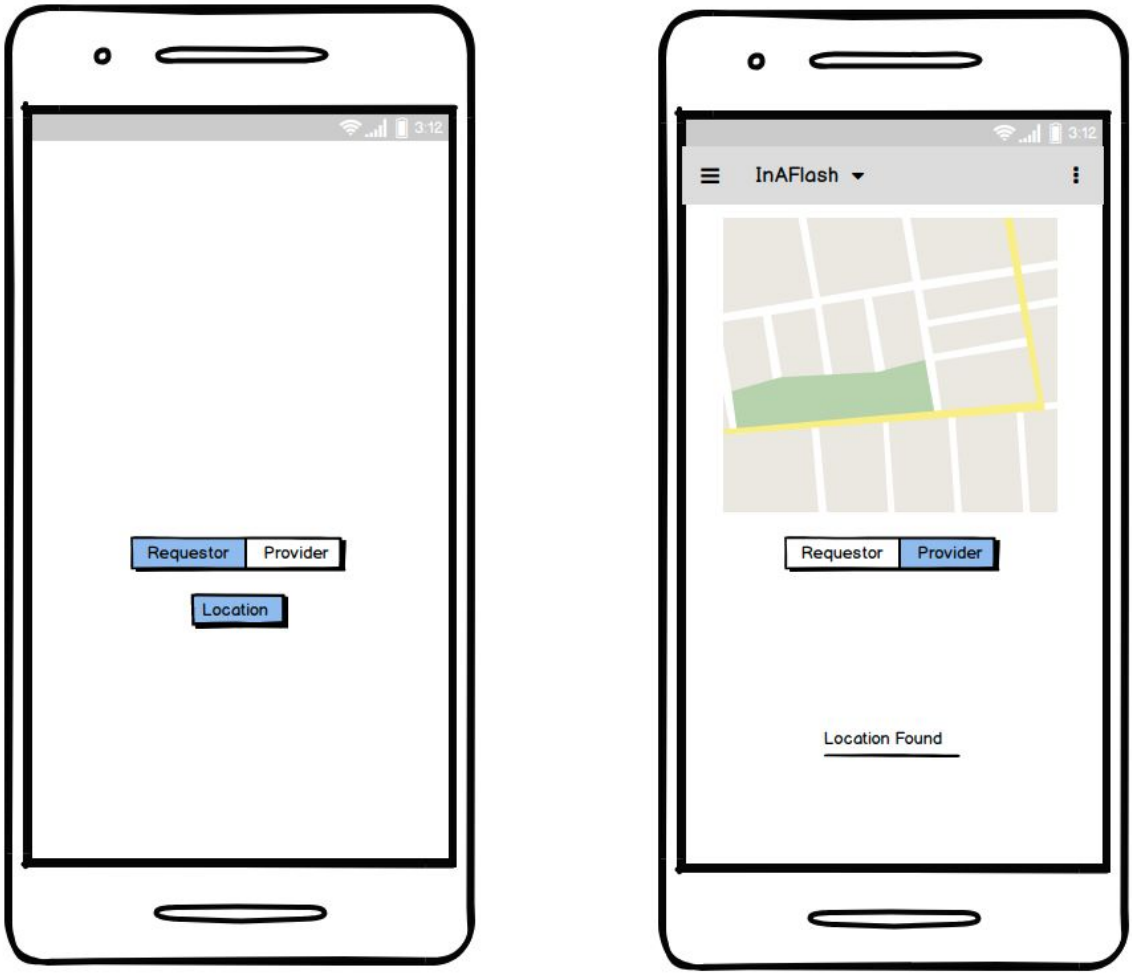
Initial Load

- Users must choose role of requestor or provider.



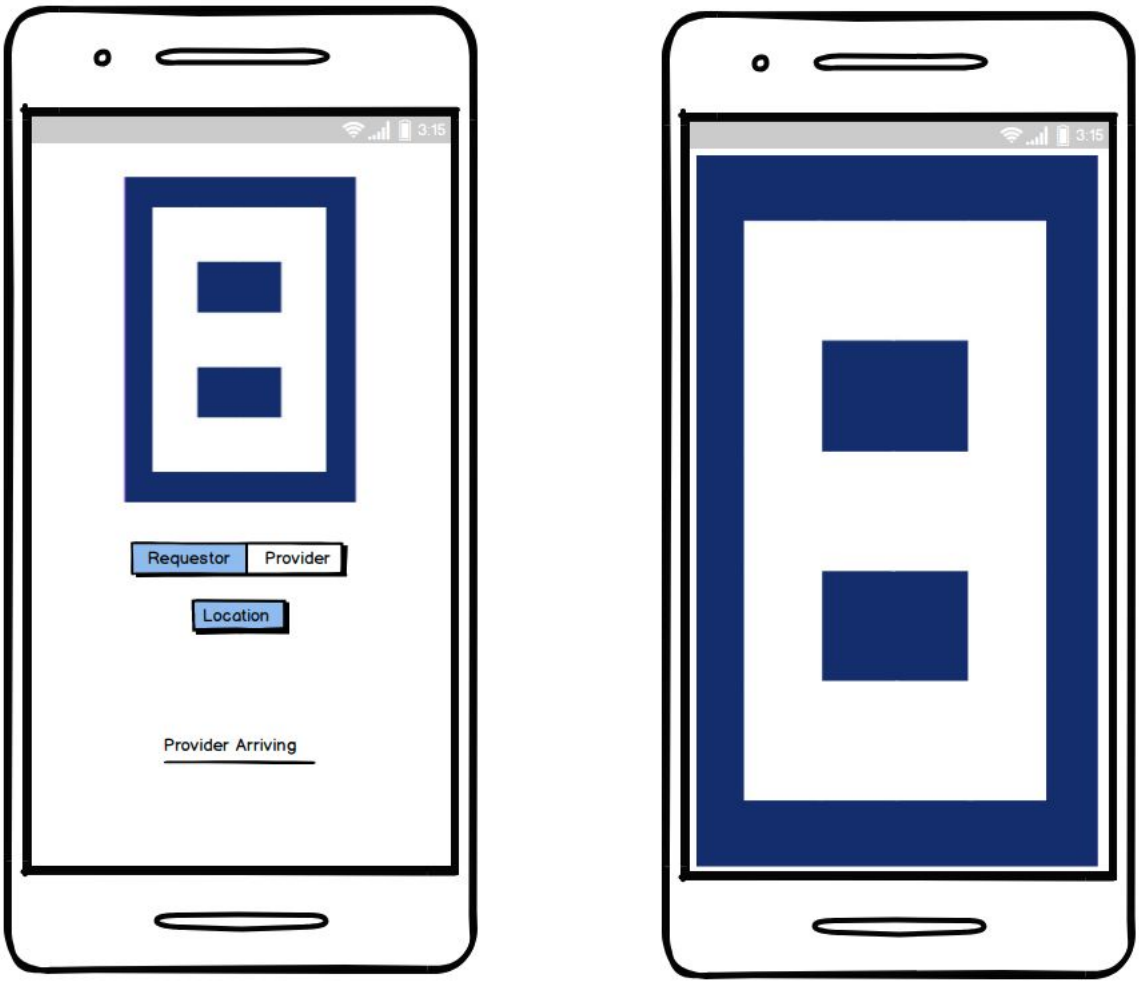
Location Set

- Requestor must set location which is sent as coordinance via FCM



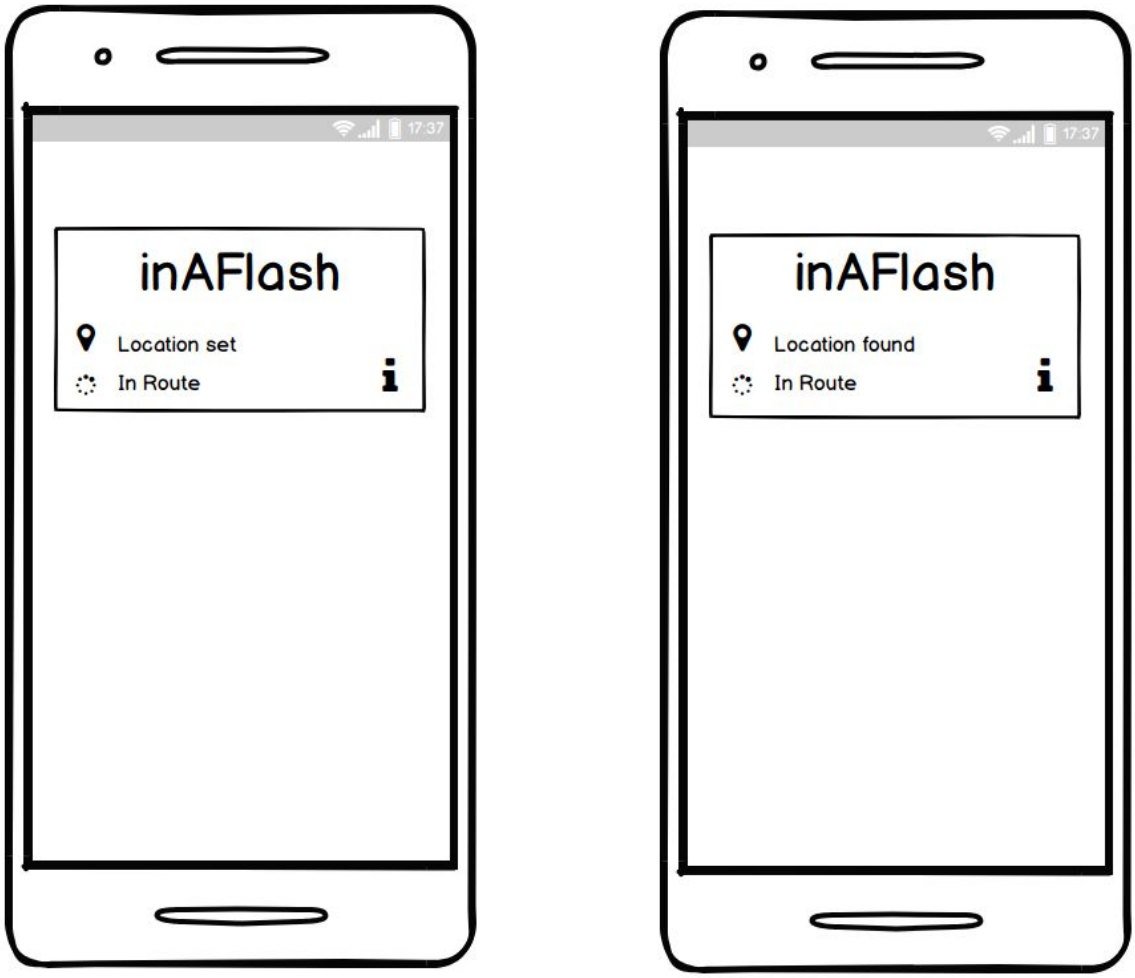
Location Found

- Provider app will create geofence via coordinance
- Maps will navigate user towards destination



Arrive & Match

- Geofence has been populated by requestor triggering event
- Card display of color/number on provider
- Full display of color/number on requestor



Widget

- Shows status of location
- Shows status of route
- Info button brings app to foreground

Rubric Specific Concerns:

- App users stable versions of AS, Gradle, and libraries

Android Studio	3.1.4
Gradle	4.6
Gradle Plugin	3.1.4
Gradle build tools	27.0.3
Minimum Sdk	19
Target Sdk	27
Appcompat-v7	27.1.1
Cardview-v7	27.1.1
Recyclerview-v7	27.1.1
design-v7	27.1.1
support-v4	27.1.1
support-v13	27.1.1
firebase-core	16.0.3
play-services-location	15.0.1
play-services-gcm	15.0.1

- ☑ App uses design library to create animations
- ☑ App validates all input from servers and users. Will log if data does not exist or is improperly formatted
- ☑ App includes support for accessibility. It has content descriptions, navigation using a d-pad, and no audio used
- ☑ All strings are stored in strings.xml and RTL switching of layouts is enabled
- ☑ App has a widget to give relevant information to the user on the home screen
- ☑ App uses location api, and firebase
- ☑ Services are imported in build.gradle of the app
- ☑ App uses device location
- ☑ App theme extends AppCompat
- ☑ App uses app bar and toolbar
- ☑ App uses standard and simple transitions between activities
- ☑ App is equipped with a signing configuration -- keystore and passwords are included in the repo. Keystore is referred by a relative path
- ☑ App dependencies are managed by Gradle
- ☑ App uses firebase realtime database with SyncAdapter
- ☑ App conforms to common standards found in the Android nanodegree general project guidelines
- ☑ App is written solely in the Java