Initial Notes:

All auto generated code such as static fragment generators and their parameters are removed.

Things that are missing:

Font families I used don't match what used on the provided screens. I was not able to determine exactly what fonts were used. I even did a check via http://www.whatfontis.com/

Custom icon for the navigation drawer was not implemented. I could not find any clear documentation on changing the new Material Navigation Drawer icon and I did not want to use the old one.

Screens

Splash(Activity)

This is the Activity class that handles the requirements for the Splash screen. Splash screens are usually shown while the application is doing some sort of loading/processing to get everything ready for the user. Since this is application is not required to do any actual time consuming loading, I created a 3 seconds long countdown timer to mimic a "loading" time.

This is also the entry point of the application. I check whether the Launch intent is loaded with any data. If the intent is fired with data loaded in it , she Splash class checks it's "scheme" for an "http" input. It indicates whether the application was launched via a potential e-mail link. If this is the case, a Google Analytics screen view is fired via the Default Tracker.

Eventually the Splash screen disappears and the user is sent to the "Core" Activity class

Core(Activity)

This class is the core of the navigation/UI flow. It implements a Navigation Drawer with Menu(Back), Home, About, and Logout options.

All Fragment views interact with this class to navigate the user. This class implements a "NavigationController" and a "FragmentNotificationListener" interface. All fragments fire their notifications through the "Core" class. When a user interacts with a fragment and is required to navigate to another fragment that is hold by the "Core" class, the fragment reaches out to the "NavigationController" which in this case is our "Core" class and the core class moves the user to the destination fragment.

This class fires heads up notifications with a custom layout. It only changes the text of the message TextView of the notifications lay out and fires them.

If the user choses to “Logout” instead of navigating somewhere else, the current user data is deleted and the user is taking back to the Splash screen as per the requirement.

Home(Fragment)

Home fragment is created , it immediately checks for two things. Whether there is an active internet connection, and whether there is a current user data available.

If there isn’t an internet connection, It fires a notification through it’s “FragmentNotificationController”. If there is no user data available then it tells it’s NavigationController to take the user to the SignUp screen so the user can SignUp and begin their session. If the user is back from the SignUp screen and/or there is already existing user data, then the application checks to see if location services are active, if not, the user prompted to the location settings screen to activate it. If all requirements are met, the user is shown their current location on Google maps as per the requirement.

SignUp (Fragment)

This class holds all the required “EditText” fields to get required user data needed to create a session. The entered e-mail is checked to match a certain pattern(Email pattern) through Android Pattern utils. If it is not a valid e-mail address then a notification fired through the fragment’s “FragmentNotificationListener” which in this case is our “Core” class.

If the user data is valid and the session creation is successful then the fragment tells it’s “NavigationController” which again is our “Core” class, to navigate to our “Home” fragment.

About (Fragment)

This fragment class simply shows 2 things, the author of the application(Me) and the application version. The layout is adjusted dynamically through “DisplayMetrics”. I initially thought supporting other screen sizes were necessary so I decided to take this route, but just left it as it is once I was told it was not a requirement.

Ancillary Classes

Session Manager

This singleton manages the current user sessions. All signups and logouts are done through the session manager. It is initialized in our application class(Ixonos) and an instance can be received anywhere in the application. Since only one user data is required be stored. I decided to simply implement it via shared preferences instead of creating a SQLite database. It holds the current user data and returns null if no user is currently signed-in.

User

This class simply abstracts an application user and holds their email, first and last names

GeoServices

This is the class that handles all location data via “GoogleApiClient”. It listens to all location updates and notifies it’s updatelisteners(OnReceiveLocationUpdate interface) which is this case is our “Home” fragment. It supports the addition of other listeners but it is not required. It gets the lat and long of the current location and converts this data to a street address via the “GeoCoder”.

GeoPoint

This class abstracts a point on a map. It holds lat , long and the street address.

WebServices

The only thing this class currently is responsible for is checking for an active internet connection. It does this via the ConnectivityManager by polling for WiFi connectivity and Mobile Cellular connection. One or the other must be available for an active internet connection to exist.

FragmentNotificationListener

Interface for a FragmentNotificationListener. A class that is responsible for any fragments that may be needing to fire notifications

NavigationController

Interface for a class that is in charge of navigating through the UI features. Fragments in this case.

ActionbarHolder

Interface for a class that will hold and change the action/tool bar titles