Test Report - GeoBus by Bus Squad (Android)

11/30/2015.

Tests Performed

As a user, I want to open the app and see a campus map and my current location so that I know where I am.

- Start GeoBus. If the user is Using any Android version under 6.0, the user will see his/her current location. If any version 6.0 or above is being used, the user will be prompted on launch to allow/deny current location permission. The user presses "Allow" and will see his/her current location.

As a user, I want to see bus stops on the map so that I know where the bus stops are.

- Start GeoBus. All bus stops used by UCSC shuttles and UCSC Metro lines are shown.

As a user, I want to tap on a bus stop and see a text box open up so that I can get more information on a particular bus stop.

- Start GeoBus. The user taps on a bus stop, which will show both its name (usually the nearest crossroad) and a box on the lower quarter of the screen with the stop's Metro buses and their scheduled arrival times.*

As a user, I want to see all active shuttles on the map so that I know if a bus will soon approach my stop.

- Start GeoBus. All active shuttles appear on the map as markers that animate when their coordinates update.

As a user, I want to see an arrow on a bus so that I know what direction it's traveling in.

- Start GeoBus. All active shuttles' markers rotate in relation to what direction they're traveling in, and each marker has an arrow pointing in that direction.**

As a Metro rider, I want to be able to see the bus schedule within the app so that I can plan my trips accordingly.

- Start GeoBus. Tapping on a bus stop will show Metro times.*

As a user, I want to be able to get more information on how the various shuttles work so that I'm not dependent on getting a brochure from a shuttle.

- Start GeoBus. Tap the three bar icon in the top right or drag from the left edge of the screen, which opens up the drawer, sidebar, or "hamburger" menu. The user is presented six options, four of which are "Loop and Upper Campus Info," "Night Core Info," "Night Owl Info," and "Night Owl Schedule." The first two options give shuttle behavior and times. "Night Core Info" gives Night Core shuttle behavior, and "Night Core Schedule" shows the exact schedule listed in the UCSC shuttle brochure.

As a user, I want to be able to zoom out and not be cluttered with bus stops so that I can see the map more clearly.

- Start GeoBus. Open the sidebar, tap on "Toggle Bus Stops." The user should see every bus stop disappear, and then reappear if he/she chooses to tap on "Toggle" again. Additionally, zooming out will cause bus stops to cluster together into one marker given they're close enough.***

* We only have a few bus times at the moment, but we can easily edit the database file that contains these in order to add more in the future.
** Markers initially aren't oriented, and so the direction they appear to point in is left. After the first location update, they'll begin to rotate as needed.
*** One bus stop around the east side can fail to toggle, but the clustering feature mostly takes care of this.

Problems During Testing

All of the issues we encountered were, by and large, Null Pointer Exceptions--attempting to make use of an object that has not been probably initialized. This came up most often after the addition of new functions and objects, which we would call in an incorrect order. For example, Android 6.0 devices were particularly odd to account for because of the way they handle permissions (allow/deny individual permissions based on time of need instead of all permissions on installation) versus older Android versions. Upon testing a permission prompt we implemented, we found that the prompt's asynchronous nature caused the app to attempt to draw the map with the user's location before the user ever gets a chance to accept/deny permission.

A peculiar issue came up when we introduced every bus stop that Metro lines 10 through 20 take, in which a certain stop around the east side of Santa Cruz would fail to set its visibility upon the user toggling bus stops. This appeared to be dependent on the bus stop JSON file's organization, as reordering stops would cause another stop within the same area of the file to fail to toggle. This issue, although never directly addressed, had been mostly taken care of with our implementation of bus stop clustering.

We attempted to get bus routes to show upon selecting an active bus marker, but this proved to be unwieldy and produced wrong results. Through the use of polylines, we intended to draw a line, following the road from a bus to its next stop or even its final destination. However, because the polylines would only draw directly from the bus to its next stop in a straight line, many coordinates along the roads would have to be hardcoded in insofar as the method we were attempting. With the little time we had remaining, we decided to scrap the feature in lieu of developing and testing other features.

While testing the bus schedule fragment, we found a (somewhat humorous) problem in which rotating the screen with a bus schedule fragment open would cause it to constantly

grow in size on each rotation until the app could no longer handle its size and crashed. We ended up circumventing this issue by disabling the user's ability to rotate the screen while viewing the main maps activity.

Another somewhat humorous issue came to light when we implemented directional markers. To do this, we took replaced the default marker icon with our own colored drawables. Upon pointing the bus marker objects to these drawables, we found that scaling was off, and each bus's marker took up roughly 20% of the screen--an absurdly large size. We addressed this by calling a specific bitmap function that rescales drawables.

While implementing the back button's behavior in regards to the sidebar, we found that the schedule fragments did not handle the same button elegantly--the button would close the entire application. We fixed this by adding fragments to a stack (a fragment-specific structure called a backstack), and then having the back button pop the entire stack in order to close the fragment.

With the connection toasts on app launch, we found that they would seemingly persist (in reality, they were just triggering at a staggeringly fast pace) if the user opened the app with no active data connection. This was circumvented with the ability to "cancel" a toast, but this caused a secondary issue in Android 6.0+ devices where, if the app prompted the user for permission (which is done in a different thread), cancelling a toast would trigger a Null Pointer Exception. This was circumvented with a simple try/catch block.

The app used to crash upon switching from Wi-Fi to mobile data or vice-versa, given that the buses had been successfully retrieved. This was addressed with a few additions to the updateMarkers runnable, one of which is a very important call to stopBackgroundData.

Known Problems/Issues

- Crash on partial connectivity on public networks such as CruzNet (i.e., before logging into the Wi-Fi network)
- Tapping on a bus or cluster marker will open the bus schedule fragment
- When the bus schedule fragment is open, tapping elsewhere on the map will only deselect the marker and not close the fragment
- Map view will appear to vertically stretch for a moment on closing bus schedule fragment
- Not all bus stops have their relevant buses' schedules
- Minor coloring issues depending on Android build used--text color in sidebar, status bar color, etc. may not appear as intended, such as appearing black when it should be white