User Documentation:

Downloading app

1. Open up the Apple **App Store** and click on the search bar.
2. In the search bar type ***GeoBus***
3. After the results are returned click and download the app

Starting up the app

1. Click on the app and it allow it to pop up
2. Begin viewing all bus updates and locations
    a. bus locations update every 10 seconds

App Features:

1. See your location on the map
    a. Your current location is represented by a tiny blue pulsating dot
2. See bus's current location in real time (ten second updating intervals)
    a. Orange dots indicate that the bus is an inner loop (clockwise)
    b. Dark blue dots indicate that the bus is an outer loop (counter clockwise)
    c. Yellow dots indicate that the bus is an upper campus bus
3. View bus schedules
4. View all on-campus bus stops in color
    a. Orange bus stops are inner bus stops (SCMTD buses 15 19)
    b. Dark blue bus stops are outer bus stops (SCMTD buses 10 12 16 20)
    c. Click on bus stop to see the name of the cross streets of that specific bus stop
5. Toggle Bus Stops
    a. The interface can seem a bit cluttered for new users. There is a big blue toggle button on the bottom of the app that when clicked will remove all of the bus stops from the screen. If clicked again, the bus stops will reappear.

Closing App:

1. Press the Home button two times quickly. You'll see small previews of your recently used apps.
2. Swipe left to find the app you want to close.
3. Swipe up on the app's preview to close it.

Developer Documentation:

This is an open source project. The project's source code can be found at https://github.com/BusSquad/geobus-ios. This is currently GeoBus v. 1.0.

This is an iOS mobile application coded in Swift 2.0 and currently works for Xcode v. 7.0.1.

Note: if you are running Xcode v. 7.1, the code will not work. We suggest downgrading your Xcode by deleting your current Xcode program (Move to Trash), going to https://developer.apple.com/downloads/, signing up as an Apple Developer, and downloading Xcode version 7.0.1.

How bus data is generated:

Campus shuttles are fitted with transmitters built into the bus's front banner signs. The transmitters read the string displaying on the banner. Examples of strings are "UPPER CAMPUS", "LOOP", "OUT OF SERVICE AT BASE", etc. Transmitters also communicate with five base stations set up around campus. Each transmitter communicates with the closest base station. This is how the location of buses is determined. This data is sent to a database that produces an XML file. The XML file is the most significant piece to this puzzle.

XML File:

The XML file can be found at http://skynet.cse.ucsc.edu/bts/coord2.xml. It displays data associated with each bus, such as its coordinates, whether it's an INNER or OUTER LOOP or UPPER CAMPUS, and the bus's ID (which is only important to the developer). Each chunk of code within <marker></marker> tags in the XML file represents data for an individual bus that is currently active on the UCSC campus. The tags for the marker bodies are self explanatory.

How the code works:

We use the Google Maps SDK to display a map centered on the UCSC campus. We also use its features to display the buses and bus stops as markers. The documentation for the Google Maps SDK can be found at https://developers.google.com/maps/documentation/ios-sdk/start?hl=en.

Swift documentation can be found at https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/.

In order to extract the data from the XML file, we parse the XML file. Swift has a built in function that does this. In order to utilize this function "`NSXMLParserDelegate`" must be included as an argument to the `ViewController` class at the top of the `ViewController.swift` file. We communicate with XML file with the following line of code:

```
parser =
NSXMLParser(contentsOfURL:(NSURL(string:"http://skynet.cse.ucsc.edu/bt
s/coord2.xml"))!)!
```

What the previous line does, is it uses `parser` as a buffer and dumps all of the XML data into it. We next must filter this data so that we only retrieve from it what is useful for us. Such as coordinates, bus direction, and LOOP or UPPER CAMPUS. We do this by creating marker objects for each bus that have fields: longitude, latitude, direction, and route. Longitude and latitude are self explanatory. Direction refers to whether it's an OUTER or INNER LOOP which means that it's either looping around campus counterclockwise or clockwise, respectively. Route specifies whether it's a LOOP or UPPER CAMPUS.

After we've built these objects, we then run a for loop that loops through these objects and uses the Google Maps SDK's functions to display the buses as markers using the lats and longs of each marker object and LOOP or UPPER CAMPUS for the titles of each marker. The type of bus icon that is used to display a bus is determined by its title. The customization of markers can be found in the Google Maps SDK documentation.

Buses aren't tracked in real time. The XML file updates every 10 seconds. This means that our code must run every 10 seconds as well. Swift has a built function called NSTimer. The line of code that runs our for loop logic every 10 seconds is

```
NSTimer.scheduledTimerWithTimeInterval(10.0, target: self, selector:
"reloadBuses", userInfo: nil, repeats: true)
```

Here, the function `reloadBuses()` repeats every 10 seconds until the app is closed. `reloadBuses()` is the function that communicates with the XML file, builds the marker objects, and runs the for loop that displays the buses on the map.

The code is commented pretty well for readability.