CSC 491 Assignment 4	Author: Krishnan Mahadevan
	Date: 25 May 2017

1. Project Title

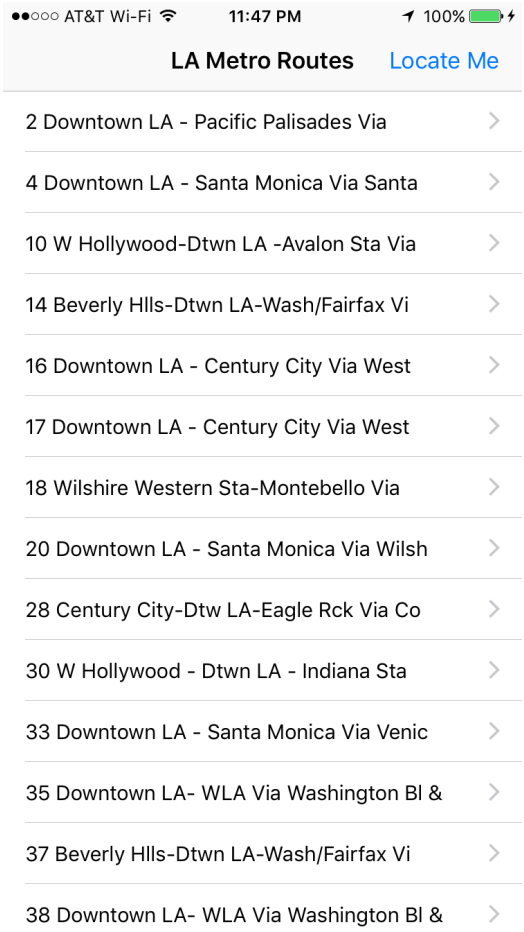
Assignment 4

2. Developer Name(s)

Krishnan Mahadevan

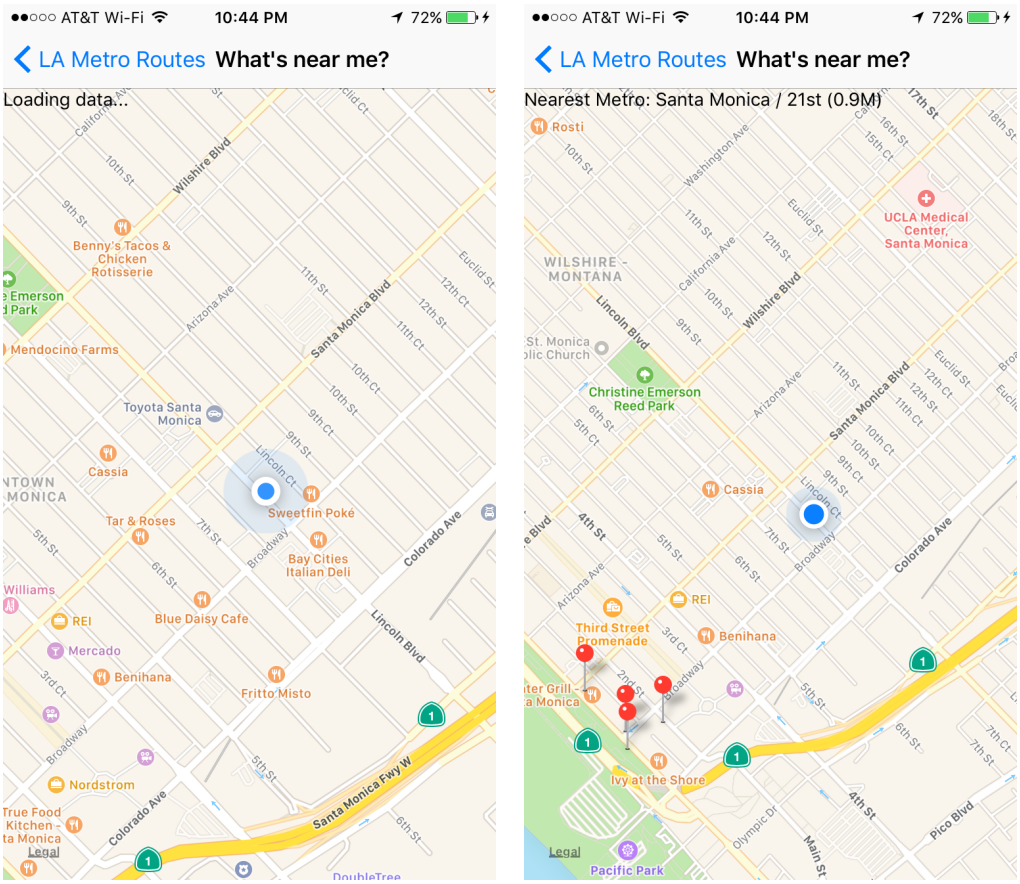
3. Project Description

Assignment 4 is an extension of assignment 2. For assignment 4, I added a new functionality to the LA Metro App called “Locate Me”. “Locate Me” offers real time data of the bus stops on the vicinity of the user. Users can reach this screen from the LA Metro Routes screen.



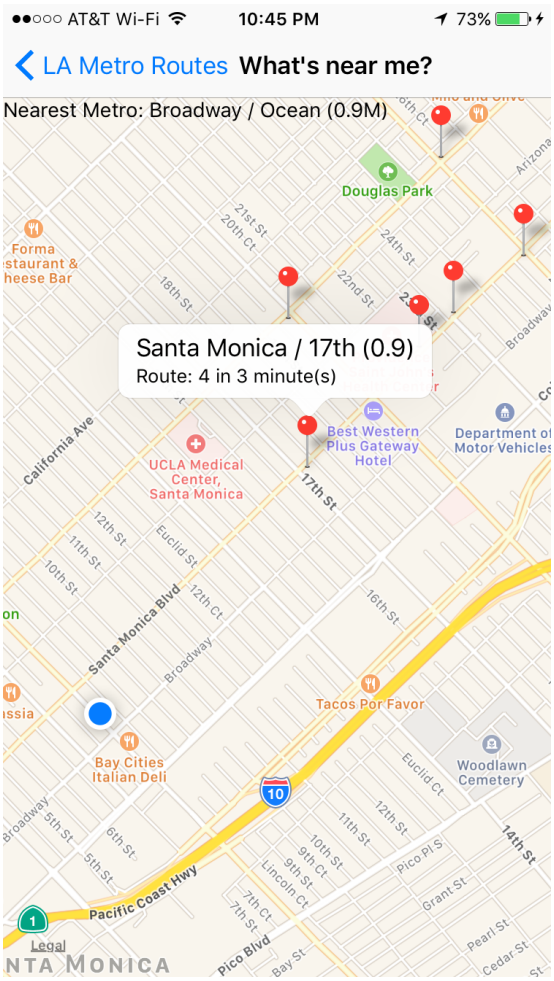
Locate Me is a data intensive screen. When this screen opens for the first time, the app preloads the available bus stops on the vicinity of the user from LA Metro API. On the initial load, users will see “Loading data” label at the top of the screen. On the subsequent loading of the screen the app picks up the bus stop information from the memory. This leads to tremendous improvement in user experience. During simulation from the .gpx file, the nearest bus stop information is loaded from phone memory instead of multiple network API calls.

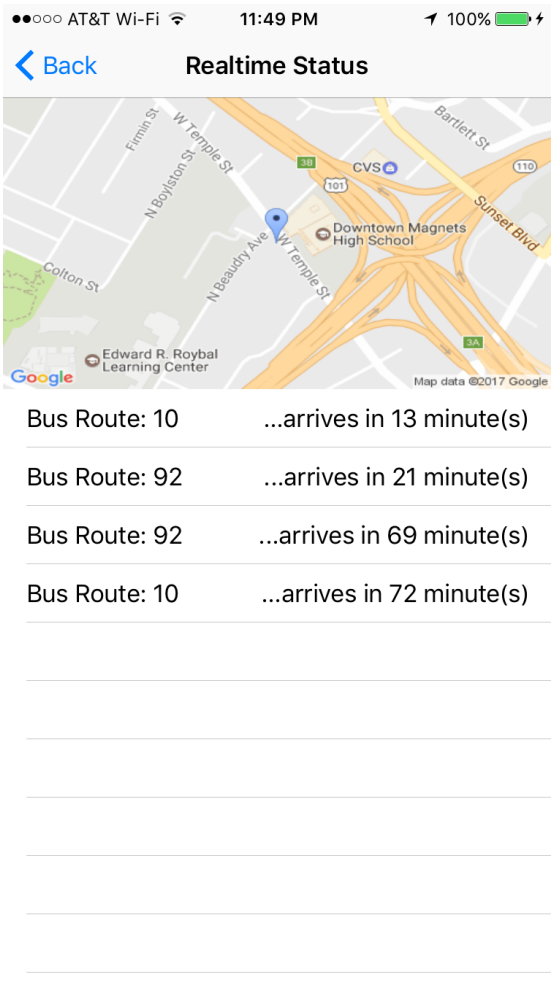
CSC 491 Assignment 4	Author: Krishnan Mahadevan
	Date: 25 May 2017



As shown on the picture above, the bus stops are shown by red annotations above. To know the bus stop name and the next bus scheduled to arrive at the location the user will have to touch the annotation. Upon touching the annotations, it shows the bus stop name and the distance of the bus stop from the user and the upcoming bus schedule. To get detailed real-time update of buses on the bus stop user will have to go on a different screen. That screen was built as a part of assignment 2. To get to real time status, user will have to go back to routes, click on route > click on bus stops on the routes > Real time status.

At any given time, user can see at least 10 bus stops in the radius of 0.5 miles. Internally, the app stores the point of refresh and updates the bus stops when the user crosses 0.5 miles from that point.





From the design perspective, the Core Location library is abstracted out into a Singleton class. This being a very expensive functionality to have, a Singleton class solves the purpose of having only one instance throughout the application.

The calculation of distance between the user's latitude and longitude and the available bus stops, is done using Haversine formula. The MetroService Class provides all the interaction of making API calls and fetching the data. The LocationSerivce class, exposes the protocol the access user's location. Errors while accessing Location is handled by simply printing it out in the output. The Location Service class has a delegate that facilitates the purpose.

While testing please note, the app takes a while to load the data from API. Once it's loaded all the subsequent operations are smooth and crisp. The project has an associated gpx file, that has a predefined route from Santa Monica to Culver City in Los Angeles.

There were a number of challenges encountered during the implementation of this assignment. The most difficult one to solve was how to access LA Metro records in real time to determine nearest bus stops in the user's vicinity and also getting all the visible bus stop status in real time. These are separate API calls and requires multiple hits to the server. I took a while to figure it out and reduce the number of API calls. The difficulty was mainly due to the non-availability of an API to determine bus stops in the user's vicinity. Hence, I had to write a custom logic to determine distance on a radius of 1 miles of the user's location.