CS 4768 Project Report

Submitted By: Osede Onodenalore, Samuel Ash, Jeff Conway

Instructor: Dr. Yuanzhu Chen

April 11, 2017

Contents

[Overview 3](#_Toc479691338)

[Core Location 4](#_Toc479691339)

[First Tab 4](#_Toc479691340)

[updateData:(CMPedometerData \*)data 4](#_Toc479691341)

[Third Tab 4](#_Toc479691342)

[- locationManager:didUpdateLocations 4](#_Toc479691343)

[locationManager:(CLLocationManager \*)manager didFailWithError:(NSError \*)error 5](#_Toc479691344)

[(void)getWeatherAtLocation:(NSString \*)lat Lon:(NSString \*)lon 5](#_Toc479691345)

[getDataFromURL:(NSURL \*)url 5](#_Toc479691346)

# Overview

For our project, we created a walking app called Commute Buddy. It is a three-tabbed application that is used for walking metrics, mapping your current location and displaying weather information. When launched the app first asks you for permission to use your current location, and for permission to use core motion data.

The first tab displays all walking metrics such as session time, the number of steps taken, the distance traveled in meters, the average number of steps per second, your current speed in meters per second, the average speed and maximum speed. There is also a reset button to reset the current session and set all these values mentioned back to zero. It does this all by using the CoreMotion and CLLocation framework.

The second tab is a map that shows your current location using the geolocation feature from the CLLocation framework. The MapKit framework is used as well to display the map. The application will also draw a line corresponding to everywhere that the user has traveled while using the application. This was done using reference from the BreadCrumb library.

The third tab displays your current weather information. It does this by using again the CLLocation framework. In addition, we are also using a weather API called OpenWeatherMap (openweathermap.org). This tap displays information such as the current temperature, the daily high and low temperature, the humidity, the air pressure, the wind and direction, and snowfall and rainfall totals. We are accessing this information by using the coordinates for latitude and longitude from the CLLocation framework.

# Core Location

As mentioned before, our application has three tabs. Each tab has its own ViewController. Each ViewController initializes and allocates an instance of CLLocationManger in the “viewDidLoad” method. As well in the viewDidLoad method, various values such are set such as, requests for authorization, desired accuracy, and to tell the manager to start updating location.

# First Tab

To keep track of the walking metrics, we make use of the CMPedometer class from the Core Motion framework. This class is initialized and allocated in the viewDidLoad method.

## updateData:(CMPedometerData \*)data

This method is called every time that that CMPedometer receives an update. Each time a variable for our different metric types (steps, distance, etc.)

# Third Tab

To keep get the weather information, we are getting the current latitude and longitude of the location from the instance of the CLLocationManger. We then use CLGeocoder to reverse-geocode and get the city name based on the coordinates and pass the coordinates as varaibles to a method to get the weather information.

## - locationManager:didUpdateLocations

\*\*\*\*\*\* Reword This\*\*\*\*\*\*\*\*\*\*\*

This is called everything time the location manager receives an update. It gets the most recent location and creates an instance of CLLocation. The longitude and latitude of the location are used in the MapKit method reverseGeocodeLocation. The city named is derived from this where the city name, latitude and longitude are all passed to the method getWeatherAtLocation

## locationManager:(CLLocationManager \*)manager didFailWithError:(NSError \*)error

This delegate method is used when there has been an error in attempting to retrieve the location value. When this occurs, a UIAlertView is created and displayed to the user saying: “Could not get your location.” If this happen, the user should verify that the location services are enabled, and try again.

## (void)getWeatherAtLocation:(NSString \*)lat Lon:(NSString \*)lon

This is a method that we defined. Here, we created the URL that is needed for an NSURLSession. The variable apiCall is an NSString that we generate to use for the NSURL. The String has a base for of “http://api.openweathermap.org/data/2.5/weather?lat=”. From there we append to the NSString the passed parameter value of the latitiude, then append on “&lon=” and the passed parameter value of the longitude. We then append on “&units=metric&appid=” to indicate that we want to use the metric system for getting our weather information. Finally, we then append on our API Key that we received from the website. For example, to get the weather information for Toronto an example URL would look like:

http://api.openweathermap.org/data/2.5/weather?lat=43.6426&lon=79.3871&units=metric&appid= 9ee3e4133c207d8258520dbdff88ec66

We then call the method getDataFromURL using the generated URL.

## getDataFromURL:(NSURL \*)url

This is the method that we use to get our data using the mentioned generated URL. The parameter is the NSURL that is needed for calling the OpenWeatherMap API to retrieve the current weather information. A call from downloadData which is defined in the AppDelegate is called for downloading information. That data that is returned is in JSON format. We then create an NSMutuableDictionary with the use of the NSJSONSerialization class. We only iterate through each of the data points when there are no errors creating the NSMutuableDictionary. There are several data points, but we are only looking for a select few. For every intended data point in the NSMutuableDictonary, we initialize an NSString with proper formatting and pass that string to the appropriate UILabel on the main.storyboard file.