**DATA REFRESH SYSTEM USING FOURSQUARE API**

Foursquare API :  <https://developer.foursquare.com/docs/venues/search>

Android Location Manager Accuracy : <https://developer.android.com/reference/android/location/Location.html#getAccuracy()>

**Purpose:**

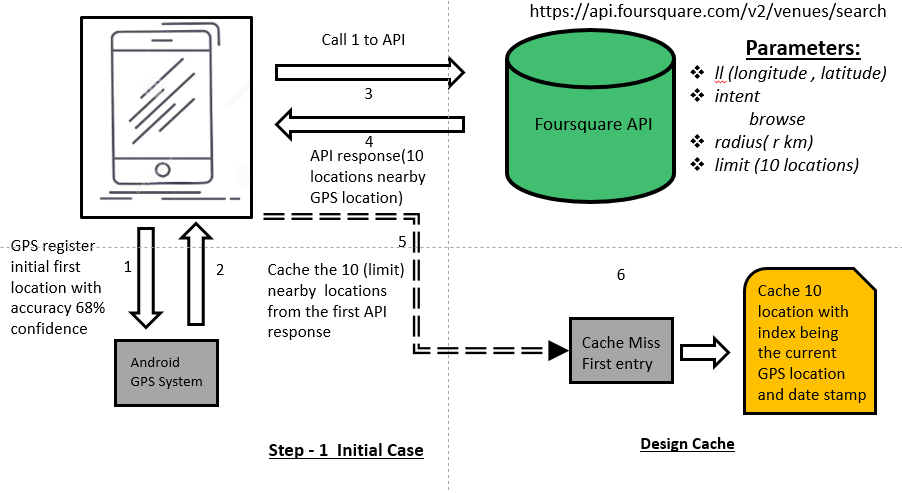
The purpose of the document is to provide a description of the design of a system that can receive a location fix (as a [latitude, longitude, accuracy] mobile GPS fix) and run an algorithm to find the actual venue a user has visited using Foursquare as a venues-database. The objective also includes minimizing the call to the Foursquare and include a way to refresh Foursquare results as data retrieved from Foursquare can only be kept for a maximum of 30 days.

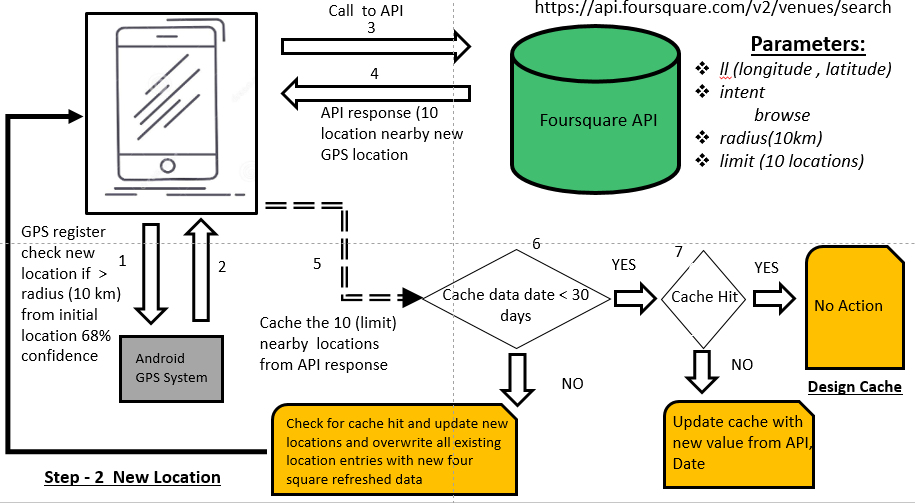
**Key Terms:**

Cache Miss, Cache Hit, Four Square API parameters (intent, radius, limit)

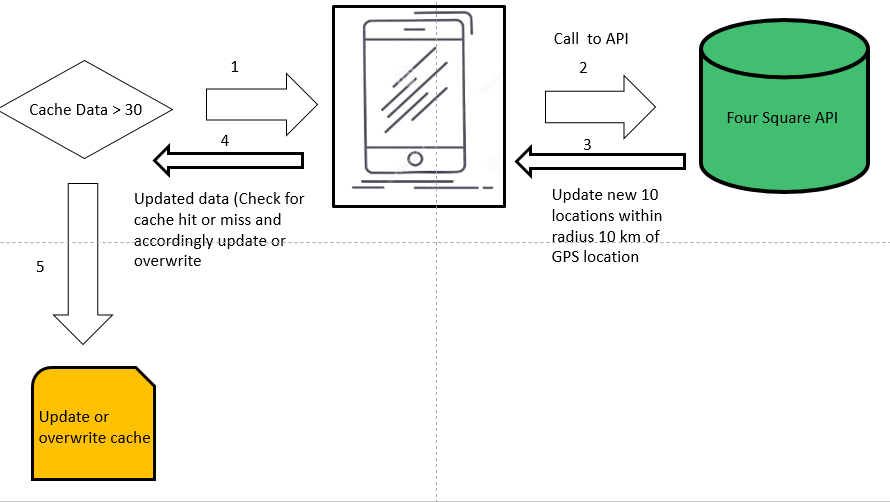
**Design Flow:**

Initial flow explains the first updating of cache (10 locations within r km->(10 km) radius) with the API values. (SLIDE -2)



Next slide explain scenario when the user moves to new location beyond the configured radius (10 KM) --- SLIDE 3

Last slide explains the cache update and overwrite when API data >30 days. SLIDE—4



**Assumptions:**

* We choose the Four-square API ‘browse’ intent, ‘fetch’ 10 locations within 10 km(‘radius’). As the GPS accuracy for a location is just 68%, Instead of pinpointing one location we choose fetch top 10 locations within a given radius (10 Km). More the radius more the number of distance comparison would be required in later stage for finding the exact location from the cache.
* Cache Hit -> If a user returns to a location previously visited within the 10km radius and already captured /cached location it is assumed be Cache Hit. Cache Hit is calculated by comparing whether the obtained GPS location lies within 10 km of any of the index values from the cache.
* Cache Miss -> If a user visits a new location beyond the 10Km radius and does not have an identical value when compared with the cache entries it is assumed to be a cache miss.
* We could use any distance algorithms to evaluate cache hit or cache miss.
* The cache is overwritten with updated details around a location after a span of 30 days.
* The GPS location entry as an index with 68 % accuracy for the cache could be approximated or singled out to an exact location by comparing the distance with each of the 10 labelled location from four square API. The shorter the distance, high likely it can be tagged to that exact location.
* After a span of 30 days there is high chance that we get updated 10 locations from four square API which could help in identifying the locations with more accuracy.
* Within the 30 days you can compare the index GPS location distance with the 10 values for identifying the most likely GPS location based on the shortest distance. This can be modified after when cache gets updated with new values from API.

**Steps:**

1. At the first step the GPS location is registered, and the first location is updated using the android location service with an estimated horizontal accuracy as the radius of 68% confidence.
2. Four square api is called passing the current *ll(latitude and longitude) ,intent (browse), fetch( radius 10 Km), limit (10 locations)* for fetching top 10 tagged location nearby .
3. The returned 10 locations are cached with index being the current gps location and most likely 10 locations within the radius of 10 km. The date when the cache entry is created is also captured. (Cache miss as initial cache is empty)
4. At the second stage the GPS location checks if we have moved out of the 10 KM radius of current location. If moved out:
5. For the current new location repeat Step 1 and 2 for fetching the details from Foursquare API.
6. If (Existing Cache date < 30)

if (Cache Hit) -> Check if location entry is existing in cache

Don’t update the new location

Else (Cache Miss) -> Location is completely new and not available in cache

Update Cache with new GPS location and 10 locations from 4 Square

Else (Existing Cache > 30)

if (Cache Hit) -> Check if location entry is existing in cache

Update overwrite the location with new locations

If (Cache Miss)

Overwrite the whole cache

Algorithm:

Each GPS location has cached 10 closest location around radius 10km.

Calculate the distance between the indexed GPS location with the nearby locations. As the points are represented as part of a sphere, we could use the **Haversine** formula which determines the great-circle distance between two points on a sphere given their longitudes and latitudes. The smaller the distance most likely the accurate location.

Also, we could use the mean distance between the GPS location and the 10 nearby points returned from the API for more accurate location.

After the span of 30 days when the cache gets updated by the new sets of API values it would help in identifying the GPS location more accurately

Example Implementation Haversan:

from math import sin, cos, sqrt, atan2, radian

# approximate radius of earth in km

R = 6373.0

lat1 = radians(52.2296756)

lon1 = radians(21.0122287)

lat2 = radians(52.406374)

lon2 = radians(16.9251681)

dlon = lon2 - lon1

dlat = lat2 - lat1

a = sin (dlat / 2)\*\*2 + cos(lat1) \* cos(lat2) \* sin(dlon / 2)\*\*2

c = 2 \* atan2(sqrt(a), sqrt (1 - a)

distance = R \* c

print ("Result:", distance)

print ("Should be:", 278.546, "km")

mpu package also uses the Haversan distance for measuring distance between 2 lat, long

import mpu

# Point one

lat1 = 52.2296756

lon1 = 21.0122287

# Point two

lat2 = 52.406374

lon2 = 16.9251681

# What you were looking for

dist = mpu. haversine\_distance((lat1, lon1), (lat2, lon2))

print(dist) # gives 278.45817507541943.

Source :- <https://stackoverflow.com/questions/19412462/getting-distance-between-two-points-based-on-latitude-longitude>