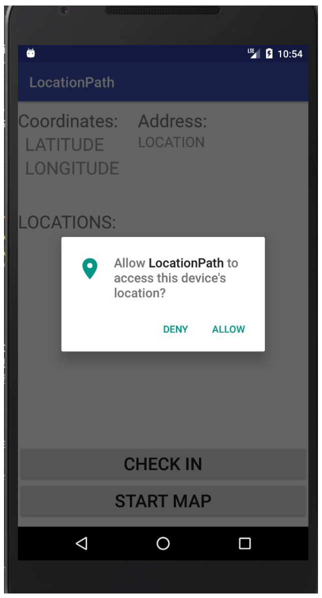
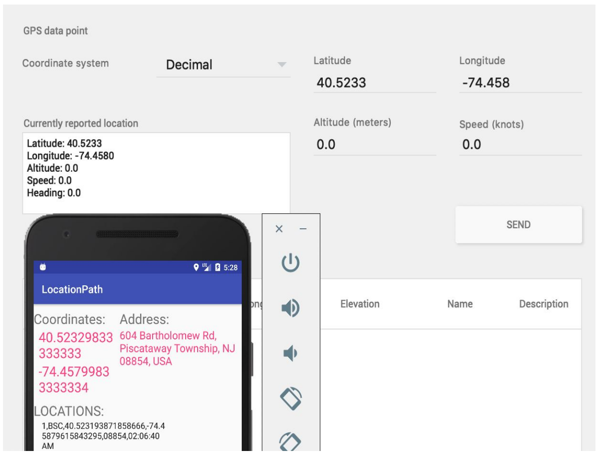
In this assignment we worked with google maps API in order to request location services for our mobile application. In this application, we explored some of the android location permission and access protocols such as ***INTERNET***, ***ACCESS_FINE_LOCATION*** and ***ACCESS_COARSE_LOCATION***. Fine and Coarse location are required to check if the user grants the application permission to access phone current location.

Permission checks are established when the app is first launched. The first thing to appeared to the user is a pop up window requesting the user to grant permission. This is done by overriding the ***AppCompatActivity's*** abstract method ***onRequestPermissionsResult***() which is triggered if Fine Location and Coarse Location are not yet granted.

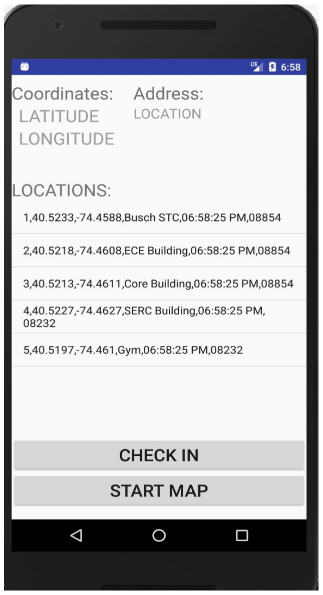

10:54
LocationPath
Coordinates: LATITUDE LONGITUDE
Address: LOCATION
LOCATIONS:
Allow **LocationPath** to access this device's location?

Once permission is granted, the application starts looking for mobile current location by using two different objects, ***FusedLocationProviderClient*** and ***LocationManager***, from the google api library. First we ask the device for any location stored in by the hardware or internal GPS. This can be seen in picture 2. Location manager constantly check for any change in location and this can be triggered by changing the coordinates manually and sending them to emulator every change will be displayed in the application TextViews. Once a location is found by LocationManager then a location will be recorded in our device. FusedLocationProviderClient will acquire every last known location. Last known location will be used for automatically check ins if no change in location has occurred in the next 5 minutes. This is done by running a second thread on the background(lines 167-172)

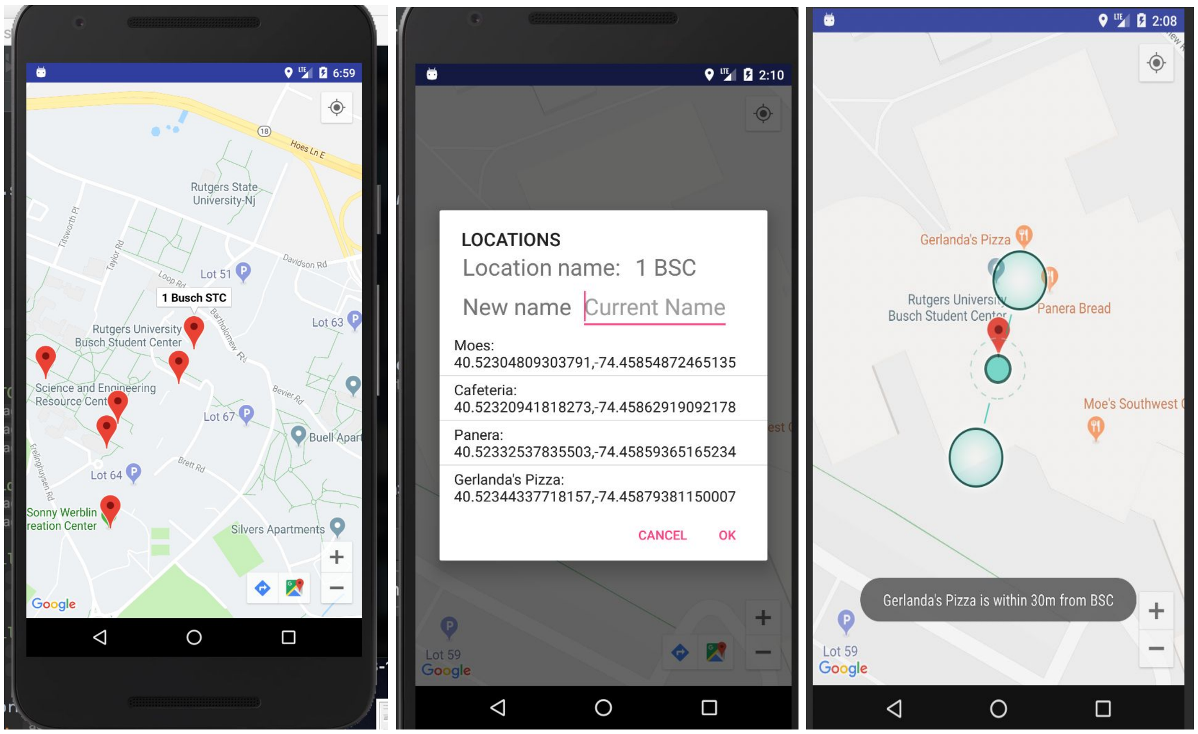

GPS data point
Coordinate system: Decimal
Latitude: 40.5233
Longitude: -74.458
Altitude (meters): 0.0
Speed (knots): 0.0
Currently reported location
Latitude: 40.5233
Longitude: -74.4580
Altitude: 0.0
Speed: 0.0
Heading: 0.0

5:28
LocationPath
Coordinates: 40.52329833333333 -74.45799833333334
Address: 604 Bartholomew Rd, Piscataway Township, NJ 08854, USA
LOCATIONS:
1,BSC,40.523193871858666,-74.45879615843295,08854,02:06:40 AM

The phone lets the user check in any location if a location is displayed, meaning the is a location and the map is running. All check in and automatically check ins will be stored in the database. Every location stored in the database will be display automatically to the user as the application is running. If a new location check in is made within 30 meters of an existing location then the new check in will be stored as related to the existing location in a new relational database table. Related locations are stored using existing location primary key as a foreign key. The key is used to query the database in order to get all locations related to a given location. By clicking any item in the listview or the marker in the map, an alert window will pop up with a list of locations related to clicked location if there exist any. Distance calculation is done by using Location object method, *distanceBetween(),* which takes coordinates between two points(lines 404-433)


6:58
Coordinates: LATITUDE LONGITUDE
Address: LOCATION
LOCATIONS:
1,40.5233,-74.4588,Busch STC,06:58:25 PM,08854
2,40.5218,-74.4608,ECE Building,06:58:25 PM,08854
3,40.5213,-74.4611,Core Building,06:58:25 PM,08854
4,40.5227,-74.4627,SERC Building,06:58:25 PM,08232
5,40.5197,-74.461,Gym,06:58:25 PM,08232

Map activity will display every location stored in the database. Database query will return a list of locations which will be looped in order to create markers for each one of them (figure 4.a). Locations within 30 meters of any location will not be displayed, instead they will be listed by clicking on the marker of a given location on the map (figure 4.b). The user is allowed to create new markers in the map and these will be stored in the database. If a new marker is within 30 meter of an existing location this will be stored as related to existing location and a Toast message will pop up to let the user know as shown in (figure 4.c).


6:59
Hoes Ln E
Rutgers State University-Nj
Davidson Rd
Lot 51
1 Busch STC
Lot 63
Rutgers University Busch Student Center
Science and Engineering Resource Cent
Lot 67
Buell Apart
Lot 64
Sonny Werblin Recreation Center
Silvers Apartments
Google
2:10
Lot 59
Google
2:08
Gerlanda's Pizza
Rutgers University Busch Student Center
Panera Bread
Moe's Southwest
Lot 59
Google

LOCATIONS
Location name: 1 BSC
New name Current Name

Moes:
40.52304809303791,-74.45854872465135

Cafeteria:
40.52320941818273,-74.45862919092178

Panera:
40.52332537835503,-74.45859365165234

Gerlanda's Pizza:
40.52344337718157,-74.45879381150007

CANCEL OK

Gerlanda's Pizza is within 30m from BSC

Figure 4

For this assignment, it was priority to get an API key to get access to google service. This assignment was conducted strictly by following the google API documentation and using all the dependencies necessary to work with different google libraries. This assignment let us explored some of the most useful RESTFul services such as Google API. We learned that android needs permission in order to get access to users device location data and how to deal with asynchronous calls by the Google API methods. The most challenging parts were to synchronise calls in order to display current location to the user as FusedLocation and LocationManager calls were made in order to get device locations. We found that this calls do not wait in order to not block and hence it was quite difficult to get around it.