**Research Objective: To learn enough about the Navigine engine to be able to implement the navigation system in our app.**

Essential Classes:

* Beacon
* Location
* LocationView
* NavigationThread
* LocationPoint
* RouteEvent
* RoutePath
* SubLocation

The essential classes are the classes that we will absolutely need in order to make the app run. The beacon class is a database container for the beacons. It has the ability to save the name, coordinates, uuid, and location of the beacons. The uuid is a unique identifier for each beacon. These are built into the beacon itself. The name for the beacon is a unique name that we can assign.

In order to use the class location we must first load a class from a file on the navigine server. This file is connected to our account that we registered with. Once the file is pulled it is saved in a directory in the phone for later use. The Location class saves the directory of this file as archiveFile. It also keeps track of a unique identifier for each location called id as well as the location’s sublocations. The methods that this class provides us with is getBeacon(), getVenue(), getVenueCategory(), and getZone() which all return a unique id for the beacon, venue, venue category, or zone. Zones are sublocations of a sublocation and a venue is a point of interest in the sublocation.

The LocationView class is probably the most important class that we need to utilize. It loads the map into our application so that the users can view our maps. It allows our users to interact with our maps, zoom, scroll, and click through the maps. The functions that deal with the coordinates are getAbsCoordinates(), getAbsLengthX(), and getAbsLengthY(). They return exactly what you’d think they do. Since every phone is different there are quite a few functions that deal with setting up the UI of the map. GetScreenCenter(), getScreenCoordinate(), getScreenLengthX(), getScreenLengthY(). These all return floats. To load the map into the view we need to use the loadSubLocation() function which returns a boolean based on whether or not the sublocation is in the Location.

The NavigationThread class works with the beacons in order to get a triangulation on the devices. It checks whether the locations are loaded correctly with hasLocation returning a boolean, and it gets the location with getLocation. We can also draw paths with the method makeRoute. Two points must be passed into the method, the starting point and the ending point. The LocationPoint class will keep track of these points. We can use the method getGraphDescription to pull down the graph so that we can iterate through the nodes with the Dijkstra algorithm.

To store the routing events we use the RouteEvent class. This class is used for when the user decides on a room to go to. It keeps track of whether the user is turning right, left, and keep track of the distance that they are from the starting position. The actual route path is stored in the RoutePath class. When a user selects the route that they want to take this stores the route and paints the pathway to the map. It also keeps track of the length of the path with the integer length, which is based in meters.

The SubLocation class is where the magic happens. It loads the picture of our map to the LocationView with getBitmap(), getPicture(), getSvgImage(), getPngImage(), or getJpgImage() depending on what type of map our map is saved as. It gets the gps coordinates of our map with getGpsCoordinates which returns an array of doubles that contain the x and y coordinates of the sublocation. GetBeacon, getVenue, and getZone get all of the sublocation’s beacons, venues, and zones returning those objects when passed the unique id identifier.