Navigation Component Description

The navigation component consists of assisting the user in traversing an indoor location to a given destination.

**Input:**

The input to the Navigation component will be the destination given verbally by the user, which will be checked against the locations index.

**Output:**

The output to the Navigation component is the instructions from the current location to the destination that are relayed to the user (in the GUI component) that help them get to their destination. This consists of pathing through building layouts, obstacles, and turns.

**Implementation:**

The app will use a combination of phone sensors and location data to find the most optimal route of a given destination. The app will then estimate the time it will take to get to this destination by using distance metrics and relay this to the user. A turn will be any part of the traversal that meets a certain angle threshold from the current position i.e. 90 degrees is an obvious turn. These turns will then be relayed to the user ahead of their arrival so they can ready themselves for it. When an obstacle is detected within a certain radius of the user t is relayed with instructions on how best to avoid it, by using the positioning of the object in relation to the walls. All feedback given and received from the user will be based on vocal recognition. The process will go like this:

1. User enters destination
2. User is prompted on the next instruction and for the duration of time it will take to get there
3. If an object or turn is upcoming, notify and the recommended course of action
4. Repeat steps 2-3 until the destination is reached.

**Validation**

The app will be tested based on its ability to complete the following actions:

* Accept a destination
* Compare destination to index of locations
* Find the optimal route to destination
* Generate an accurate ETA
* Recognize and notify of obstacles
* Recognize and notify of turns

Unit tests for these can be generated by comparing the expected output to the actual output. For instance, we can generate a test that has a mock input for a destination, and check if this destination is properly received by checking the function in our app that returns the user input. If the output matches, we know it was stored correctly.