1.3 - Identifying test approaches

At the system level, I've chosen:

1. No response for an endpoint should take more than 30 seconds.
    1. My strategy for this is that the query that will take the longest will be the path-finding one, therefore this can be tested for by running queries with larger numbers of medDispatches

At the integration level, I've chosen multiple related flight path requirements:
3. A drone flight should deliver all medDispatches assigned to it (if possible)
1. A "delivery" counts as a drone hovering (The same location twice in the flight path) within 0.00015° of the delivery location.
4. Ensure flight paths begin and end at the same service point.
5. Ensure the flight paths are otherwise correct

My strategy for these is to:

1. Analyse returned paths and ascertain that all delivery locations are included
2. Analyse returned paths and ensure that the start and end position is within 0.00015° of the same service point.

At the unit level, I've chosen:
5. Drones can only move with an angle that is a multiple of 22.5°
6. Drones can only move by *exactly* 0.00015° in a given direction
7. Drones should not be able to fly over no-fly zones *(no fly zones are defined as rectangles in LngLat space)*. This especially includes corner cutting.
All of my unit-level tests are covered by following the ***Partition*** principle from Y&P chapter three, breaking down the integration level requirement of "Return an otherwise correct path". For these tests, I will analyse the individual path from A to B given by my A-star algorithm and ensure that none of these rules are broken at any given point in that path. This ensures that any larger paths built up (e.g. A -> B -> C) also abide by these rules