Running manager test documentation

We are using unit tests. We also do tests for each auxiliary function and the updateLocation procedure. Expo jest help us in test.

calculateAvg

In this test are 3 cases.

Test case no.	parameters	Expected result
1	Dist: 1km, time: 3600000ms	1
2	Dist: 10km, time: 3600000ms	10
3	Dist: 1km, time: 360000ms	10

GetCopiedLocation

In this test phase, we are checking, that last location values will be copied in the current location json. In test cases we are checking all elements of result json.

Input parameters:

Array:

```
[{"latitude":0, "longitude":0, "altitude":1, "accuracy":12, "altitudeAccuracy": 21 "heading":10, "timestamp":1000}]';
```

CurrentLocation:

```
'{"coords": {"latitude":100, "longitude":100, "altitude":100, "accuracy":100, "altitudeAccuracy" :100, "heading":100}, "timestamp":2000}'
```

Test case no.	Json element	Expected result
1	latitude	0
2	longitude	0
3	altitude	1
4	accuracy	12
5	altitudeAccuracy	21
6	heading	10
7	timestamp	2000

GetDistance

In this test, we expect results in centimeters. Let's look at the correct calculation of the distance.

Test case no.	Json element	Expected result
1	`'[{"latitude":0, "longitude":0}, {"latitude":0.00001, "longitude":0}, {"latitude":0.00003, "longitude":0}]'`	Distance between first and last element and distance between first and second element three times are equal.

GetDistanceOfLastElements

In this test phase, we examine the distances in an array of 7 elements. The distance sum of the last x elements is compared with the distance between the last and the given element. The distance increases proportionally in the steps so you have to match.

Test case no.		Test parameters	Expected result
1	from	Lat.: 0.00005, long.: 0	getDistance(testJSON[5],testJSON[6])
	to	Lat.: 0.00006, long.: 0	
2	from	Lat.: 0.00004, long.: 0	getDistance(testJSON[4],testJSON[6])
	to	Lat.: 0.00006, long.: 0	
3	from	Lat.: 0.00003, long.: 0	getDistance(testJSON[3],testJSON[6])
	to	Lat.: 0.00006, long.: 0	
4	from	Lat.: 0.00002, long.: 0	getDistance(testJSON[2],testJSON[6])
	to	Lat.: 0.00006, long.: 0	
5	from	Lat.: 0.00001, long.: 0	getDistance(testJSON[1],testJSON[6])
	to	Lat.: 0.00006, long.: 0	
6	from	Lat.: 0, long.: 0	getDistance(testJSON[0],testJSON[6])
	to	Lat.: 0.00006, long.: 0	

GetHHMMSS

The time in HHMMSS format. Each digit is examined.

Test case no.	Time parameter in ms	Expected result	Recived result
1	1000	00:00:01	00:00:01
2	10000	00:00:10	00:00:10
3	60000	00:01:00	00:01:00
4	600000	00:10:00	00:10:00
5	3600000	01:00:00	01:00:00
6	36000000	10:00:00	10:00:00
7	86400000	00:00:00	00:00:00

countStops

In this test, we look at how the stop counting function works.

Test case no.	Test array	Expected result
1	[0,0,0,0,0,0]	0
2	[0,1,1,0,1,1]	1
3	[0,1,0,1,0,1]	2
4	[0,1,0,0,0,1]	1

toFixing

In this test, we look at how the flooring function works.

Test case no.	Parameter	Expected result
1	Num: 1.125, dec: 0	1
2	Num: 1.125, dec: 1	1.1
3	Num: 1.125, dec: 2	1.12
4	Num: 1.125, dec: 3	1.125

updateLocation

This is the most important test. The test consists of 6 parts. Test check the array length, the results of the auxiliary variables, and the auxiliary functions.

Auxiliary array

Now, the growth of the array is observed. When the function is called with a data set, the size of the array increases by one.

Distance

In this test, we look at whether the distance increases proportionally.

Useless data drop

The first 2 data in the test environment are useless. Since it affects the measurement, we are the first few items discarded.

TooSlow

Slow travel speed is indicated by the application. In this phase, we test whether it really signals.

RunCoordinates array

Size should increase, similar to an auxiliary array. Testing tactic is same as in the first case.

Conclusion

Testing was done and it worked for each case. We tried to perform the test to the most critical values.