**Cleaning Service Functional Specs:**

**Introduction**

You will test a service that navigates a imaginary robotic hoover (much like a Roomba) through an equally imaginary room based on:

- room dimensions as X and Y coordinates, identifying the top right corner of the room rectangle. This room is divided up in a grid based on these dimensions; a room that has dimensions X: 5 and Y: 5 has 5 columns and 5 rows, so 25 possible hoover positions. The bottom left corner is the point of origin for our coordinate system, so as the room contains all coordinates its bottom left corner is defined by X: 0 and Y: 0.

- locations of patches of dirt, also defined by X and Y coordinates identifying the bottom left corner of those grid positions.

- an initial hoover position (X and Y coordinates like patches of dirt)

- driving instructions (as cardinal directions where e.g. N and E mean "go north" and "go east" respectively)

The room will be rectangular, has no obstacles (except the room walls), no doors and all locations in the room will be clean (hoovering has no effect) except for the locations of the patches of dirt presented in the program input.

Placing the hoover on a patch of dirt ("hoovering") removes the patch of dirt so that patch is then clean for the remainder of the program run. The hoover is always on - there is no need to enable it.

Driving into a wall has no effect (the robot skids in place).

**Goal**

The goal of the service is to take the room dimensions, the locations of the dirt patches, the hoover location and the driving instructions as input and to then output the following:

The final hoover position (X, Y)

The number of patches of dirt the robot cleaned up

Your goal is to verify whether the provided implementation behaves according to specification.

**Service specification**

Input

Program input will be received in a json payload with the format described here.

Example:

{

"roomSize" : [5, 5],

"coords" : [1, 2],

"patches" : [

[1, 0],

[2, 2],

[2, 3]

],

"instructions" : "NNESEESWNWW"

}

Output

Service output is returned as a json payload.

Example (matching the input above):

{

"coords" : [1, 3],

"patches" : 1

}

Where coords are the final coordinates of the hoover and patches is the number of cleaned patches.

Test the endpoint:

curl -H 'Content-Type: application/json' -X POST -d '{ "roomSize" : [5, 5], "coords" : [1, 2], "patches" : [ [1, 0], [2, 2], [2, 3] ], "instructions" : "NNESEESWNWW" }' http://localhost:8080/v1/cleaning-sessions

**Test Plan:**

**Assumptions**:

1. The service is built and is up and running at the given port.

2. Room has walls, so hoover's input coords outside room coords is invalid. Service should return specific

error code if the initial hoover position is outside the valid room coordinates. This can help indicate to

the user or calling program that there is an issue with the input.

3. Service should return appropriate messages in the response body when input parameters are missing/invalid.

This helps the consuming application to understand what parameters are invalid.

4. If service is stopped/not running, we want to check at the suite setup and fail quickly without having to run any tests. Doing this in line #14 in CleaningServiceTests.robot

Test Cases and execution status:

| Test Case Number | Description | Execution Status | Bug ID |
| --- | --- | --- | --- |
| 1 | Verify cleaning service returns 200 when valid request payload is sent | Pass |  |
| 2 | Verify cleaning service returns 400 when no payload is sent in the request | Pass |  |
| 3 | Verify cleaning service returns 400 when roomSize is invalid | Fail | PltSci\_ClnSvc\_002 |
| 4 | Verify cleaning service returns 400 when roomSize is missing | Pass |  |
| 5 | Verify cleaning service returns 400 when coords are invalid | Fail | PltSci\_ClnSvc\_002 |
| 6 | Verify cleaning service returns 400 when coords is missing | Pass |  |
| 7 | Verify cleaning service returns 400 when patches is invalid | Fail | PltSci\_ClnSvc\_002 |
| 8 | Verify cleaning service returns 0 cleaned patched when patches is empty | Pass |  |
| 9 | Verify cleaning service returns 400 when instructions are invalid | Fail | PltSci\_ClnSvc\_002 |
| 10 | Verify cleaning service returns 0 cleaned patches when no instructions are sent | Pass |  |
| 11 | Verify cleaning service returns 0 cleaned patches when driving directions don't travel patches of dirt | Pass |  |
| 12 | Verify cleaning service returns 1 cleaned patch when hoover is at corner spot which is also dirt spot and driving instructions hit the wall | Pass |  |
| 13 | Verify cleaning service returns 0 cleaned patch when hoover is at corner spot which is not a dirt spot and driving instructions hit the wall | Pass |  |
| 14 | Verify cleaning service does not cache cleaned spots from previous cleaning sessions when these spots are traveled during subsequent requests | Fail | PltSci\_ClnSvc\_001 |
| 15 | Verify cleaning service returns 400 when coords placed exactly at the room coordinates | Pass |  |
| 16 | Verify cleaning service returns 400 when coords are outside the room coordinates | Fail | PltSci\_ClnSvc\_003 |
| 17 | Verify cleaning service cleans patches of dirt at each of 4 corners of the room | Fail |  |
| 18 | Verify cleaning services return total patches when every location has dirt | Pass |  |

**Bugs:**

1. Previously cleaned patches are cached.

Bug Id: PltSci\_ClnSvc\_001

Bug Description: Cleaned patches are being cached and accounted for as long as they are being traveled in requests.

Bug Severity: High

Steps to Recreate:

Request1:

input "roomSize": [5, 5], coords = [2, 3], patches = [[2,3], [3, 3]], instructions = EW

Expected Response = coords = [2,3], patches = 2

Request 2:

input "roomSize": [5, 5], coords = [2, 3], patches = [[4,3]], instructions = WEEE

Expected Response = coords = [4,3], patches = 1

Actual Response = coords = [4,3], patches = 3 (The 2 patches [2,3], [3,3] that are cleaned in previous requests are cached and accounted for as they are in the driving directions)

After restarting service, request 2 resulted in:

Response = coords = [2,3], patches = 1

2. Based on Assumption #3 above, When input parameters are invalid, service response should return appropriate messages in the response body.

Bug Id: PltSci\_ClnSvc\_002

Bug Description: When input parameters are invalid, service response should contain detailed error message.

Bug Severity: Medium

Steps to Recreate:

1. Send a request with "roomSize": [0, 0], coords = [2, 3], patches = [[4,3]], instructions = WEEE

Response code is 400 - Bad Request. But the response body is empty. Include a response body with a message stating what parameters are invalid and why.

See test case 3, 5, 7 and 9 they all have this issue.

3. Based on Assumption #2 above, Service should return a 400 error with appropriate error message when hoover coords are outside room coordinates.

Bug Id: PltSci\_ClnSvc\_003

Bug Description: When hoover coords are outside the room coordinates, service should return a 400 code with error message in response body.

Bug Severity: Medium

Steps to Recreate:

1. Send a request with "roomSize": [5, 5], coords = [8, 10], patches = [[2,3], [3, 3]], instructions = EW

Response code should be 400, with a response body.