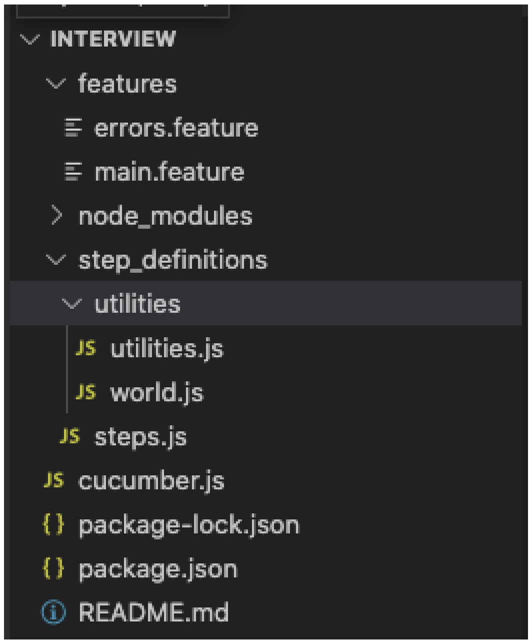
Coding challenge report

Dillan Butler

Test structure

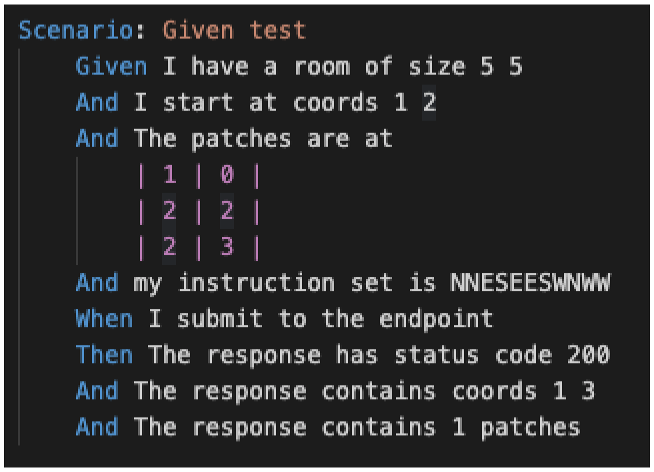


```
INTERVIEW
  features
    errors.feature
    main.feature
  node_modules
  step_definitions
    utilities
      utilities.js
      world.js
    steps.js
  cucumber.js
  package-lock.json
  package.json
  README.md
```

For the structure of my tests I went with a standard cucumber structure. I made two feature files, one for testing the successful and another for testing the error paths, the program only has one feature so this was just done for readability purposes.

Only one steps file was needed as there is only one feature, normally I would create additional step files for different endpoints, features or pages (in the case of UI testing)

I created a utilities file and a world.js file as well. The utilities file only has one function (an HTTP post function) but this could be expanded to other HTTP functions as well as any other helper functions that would be needed across multiple step files.



```gherkin
Scenario: Given test
    Given I have a room of size 5 5
    And I start at coords 1 2
    And The patches are at
        | 1 | 0 |
        | 2 | 2 |
        | 2 | 3 |
    And my instruction set is NNESEESWNWW
    When I submit to the endpoint
    Then The response has status code 200
    And The response contains coords 1 3
    And The response contains 1 patches
```

For the gherkin syntax I went with an imperative style instead of declarative. In general, I prefer imperative for API testing and declarative for UI testing. This is because imperative gives you more granular control at the expense of readability, which can be useful in API testing but when it comes to UI testing, I prefer to relate steps to business logic rather than components on the page.

Assumptions

Beyond the given requirements I made a couple assumptions

1. All inputs (roomSize, coords, patches and instructions) are required
2. A room size of at least (1,1) is required
3. Room size lengths must be a non negative integer
4. Starting coordinates both must be non-negative integers
5. Patch location coordinates both must be a non-negative integers
6. Starting coordinates must be inside the room size
    a. 0 <= starting_x < size_x && 0<=starting_y < size_y
7. Patch location coordinates must be inside of the room size
    a. 0 <= patch_x < size_x && 0<=patch_y < size_y for all patches
8. Instruction set must be a string only containing the characters N, S, E and W
9. Each run is separate, that is to say all data (room size, patch locations, starting locations) is cleared on a new run

Outstanding questions

Beyond the given requirements and assumptions that I made, I had some questions about functionality

1. Are my assumptions correct?
2. What is the expected result when there are two patches at the same location?
3. What is the expected result when the instruction set is an empty string
4. What is the expected result when the starting location is the same as the location of a patch?

Bugs found

1. Y starting coordinate does not return an error when set to a negative number
    a. Scenario that fails
        i. Scenario Outline: Invalid starting coordinates <starting_x>, <starting_y>
    b. Expected: When the starting Y coordinate is set to a negative number, a 400 response is returned from the server
    c. Actual: When the starting Y coordinate is set to a negative number, a 200 response is returned from the server
    d. Note: Appropriate error is returned when just the starting x is a negative number
2. When room size is not set, a 500 error is returned
    a. Scenario that fails

      i. Scenario: Empty room size
   b. Expected: When the room size is not set, a 400 error should be returned.
   c. Actual: When the room size is not set, a 500 error is returned
   d. Notes:
      i. A 500 error indicates a server issue, a 500 error should not be returned based on invalid user input.
3. No error returned when room size x or y is negative
   a. Scenario that fails
      i. Scenario Outline: Invalid room size <room_x> <room_y>
   b. Expected: When the X or Y for the room size is set to a negative number, a 400 response is returned
   c. Actual: When the X or Y for the room size is set to a negative number, a 200 response is returned
4. No error returned when patch y is a negative number
   a. Scenario that fails
      i. Scenario Outline: Invalid patch position <patch_x> <patch_y>
   b. Expected: When patch Y coordinate is negative, a 400 response code error is returned
   c. Actual: When the patch Y coordinate is negative, a 200 response code is returned
   d. Note: Appropriate error is returned when the patch x coordinate is a negative number
5. No error returned when patch coordinate is set to a non-integer character
   a. Scenario that fails
      i. Scenario Outline: Invalid patch position <patch_x> <patch_y>
   b. Expected: When patch coordinates are set to a non-integer character (for example the letter ‘A’), a 400 response code error is returned
   c. Actual: When patch coordinates are set to a non-integer character (for example the letter ‘A’), a 200 response code error is returned
6. No error returned when patch coordinate is set to a non-integer number
   a. Scenario that fails
      i. Scenario Outline: Invalid patch position <patch_x> <patch_y>
   b. Expected: When patch coordinates are set to a non-integer number (for example ’1.1’), a 400 response code error is returned
   c. Actual: When patch coordinates are set to a non-integer number (for example ’1.1’), a 200 response code s returned
7. No error is returned when patches are out of bounds
   a. Scenario that fails
      i. Scenario Outline: Patch out of bounds <patch_x> <patch_y>
   b. Expected: if a patch is placed out of bounds (patch_x > room_x or patch_y > room_y) a 400 error response is returned
   c. Actual: if a patch is placed out of bounds (patch_x > room_x or patch_y > room_y) a 200 response is returned
8. Patches are not cleared between runs

    a. Scenario that fails
        i. Scenario: Patches cleared between runs
    b. Expected: Each run is a separate event and patch locations are not saved between runs
    c. Actual: If a call is made with a patch and not cleaned up, then subsequent runs will take place as if that patch is still there causing unexpected results

9. Starting on a patch does not clear that patch
    a. Scenario that fails
        i. Scenario: Start on patch
    b. Expected: If the starting position is the same as a patch position then that patch should be cleared (if patch_x = start_x and patch_y = start_y then patches in the response should be greater than 0)
    c. Actual: If the starting position is the same as a patch position then that patch is not cleared
    d. Note: This bug is subject to my outstanding questions
10. If two patches are in the same location, only one is cleaned when the roomba passes over it
    a. Scenario that fails
        i. Scenario: Two patches in same spot
    b. Expected: If two patches exist in the same location then both should be cleaned when the roomba passes over it
    c. Actual: If two patches exist in the same location then both are not cleaned when teh roomba passes over it
    d. Note: If the roomba passes over the location twice, then both patches are cleared. This is an open question about how double patches should be handled