Test Plan Document: Rocket Jump Ninja Search Engine Automation Test

Website: Rocket Jump Ninja

API Endpoint: <https://www.rocketjumpninja.com/api/search/mice>

Programming Language: Python

**Test Scenarios**

**test\_width()**

iterates through a list of parameter sets, each representing a different test case with different width and leniency values. Test will print the acutal and also the min and max for each mice   
Test Cases:

set width 9 leniency set to 0

set width 11 leniency set to 1

set width 12 leniency set to 4

params = [  
 (**'CLAW'**, 0, **'false'**, **'false'**, **'both'**, 9, 20, 5, **'set width 9 leniency set to 0'**),  
 (**'FINGERTIP'**, 1, **'false'**, **'false'**, **'both'**, 11, 20, 5, **'set width 11 leniency set to '**),  
 (**'PALM'**, 4, **'false'**, **'false'**, **'both'**, 12, 20, 5, **'set width 12 leniency set to'**),

excpcted: all serach results to be printed, and all width values are in range

falls within the specified boundaries

falls outside the specified boundaries

**no issues found**

---------------------------------------------------------------------------------

**test\_ length()**

Each test case evaluates the behavior of the system under different conditions of length leniency, ensuring that the mouse length is within the set bounds.  
  
params = [  
 (**'CLAW'**, 0, **'false'**, **'false'**, **'both'**, 9, 20, 5, **'set length 20 leniency set to 0'**),  
 (**'CLAW'**, 1, **'false'**, **'false'**, **'both'**, 9, 19, 5, **'set length 19 leniency set to 1'**),  
 (**'CLAW'**, 4, **'false'**, **'false'**, **'both'**, 9, 18, 5, **'set length 18 leniency set to 4'**),

Test Cases:

set length 20 leniency set to 0'

set length 19 leniency set to 1'

set length 18 leniency set to 4'  
excpcted: all serach results to be printed, and all width values are in range   
falls within the specified boundaries

falls outside the specified boundaries

**no issues found**

---------------------------------------------------------------------------------

**number\_of\_buttons():**

Each test case will take a diffrent number of buttens , and leniency set to 0

To check if number of buttens is searched right

params = [  
 (**'CLAW'**, 0, **'false'**, **'false'**, **'symmetrical'**, 9, 20, 3, **'3 buttons, leniency set to 0'**),  
 (**'CLAW'**, 0, **'false'**, **'false'**, **'symmetrical'**, 9, 20, 4, **'4 buttons, leniency set to 0'**),  
 (**'CLAW'**, 0, **'true'**, **'false'**, **'asymmetrical'**, 9, 20, 5, **'5 buttons, leniency set to 0 '**),  
 (**'CLAW'**, 0, **'true'**, **'false'**, **'both'**, 9, 20, 6, **'6 buttons, leniency set to 0'**),  
 (**'CLAW'**, 0, **'true'**, **'false'**, **'both'**, 9, 20, 7, **'7 buttons, leniency set to 0'**),

3 buttons, leniency set to 0

4 buttons, leniency set to 0

5 buttons, leniency set to 0

6 buttons, leniency set to 0

7 buttons, leniency set to 0

Excpted : when wleniency set to 0, to have no issues

Test Case 2: Description: 4 buttons, leniency set to 0, Name: Zowie ZA13, Number of Buttons: Expected Buttons: 4 | Actual Buttons: 5

**issues found**

---------------------------------------------------------------------------------

**test\_wireless\_integrity()**

Each test case will check wirelss for true or false , with leniency set to High and low

Test will print if wireless value is diffrent then requsted during search  
  
params = [  
 (**'CLAW'**, 0, **'false'**, **'false'**, **'both'**, 9, 20, -1, **'Wireless false, low leniency'**),  
 (**'CLAW-PALM'**, 0, **'true'**, **'false'**, **'both'**, 9, 20, -1, **'Wireless true, low leniency'**),  
 (**'PALM'**, 4, **'false'**, **'false'**, **'both'**, 9, 20, -1, **'Wireless false, high leniency'**),  
 (**'FINGERTIP'**, 4, **'true'**, **'false'**, **'both'**, 9, 20, -1, **'Wireless true , high leniency'**),

Wireless false, low leniency

Wireless true, low leniency

Wireless false, high leniency

Wireless true, high leniency

Excpted : when wleniency set to 0, to have no issues

Test Case 1: Description: Wireless false, low leniency,Issue With Name: Lamzu Atlantis Mini, Wireless Integrity Result: expected wireless: false | actual wireless: true

**issues found**

---------------------------------------------------------------------------------

**left\_handed()**

Each test case will check left\_handed for true or false , with leniency set to low and high

Test will print if left\_handed value is diffrent then requsted during search

params = [  
 (**'CLAW'**, 0, **'false'**, **'false'**, **'both'**, 9, 20, -1, **'left\_handed false leniency 0'**),  
 (**'PALM'**, 4, **'true'**, **'false'**, **'both'**, 9, 20, -1, **'left\_handed false leniency 4'**),  
 (**'FINGERTIP'**, 0, **'false'**, **'true'**, **'both'**, 9, 20, -1, **'left\_handed true leniency 0'**),  
 (**'CLAW'**, 4, **'true'**, **'true'**, **'both'**, 9, 20, -1, **'left\_handed true leniency 0'**),

left\_handed false leniency 0

left\_handed false leniency 4

left\_handed true leniency 0

left\_handed true leniency 4

Excpted : to have no issues when set to true or false regradless of other params such as leniency or grip type

**no issues found**

---------------------------------------------------------------------------------

**test\_wrong\_values\_empty\_response():**

Tests will check for wrong and right values for &shape=XXX

params = [  
 (**'CLAW'**, 0, **'false'**, **'false'**, **'xboth'**, 9, 20, -2, **'Wrong Shape Value'**),  
 (**'CLAW'**, 0, **'false'**, **'false'**, **'symmetrical'**, 9, 20, -2, **'Right Shape Value: Asymmetrical'**),  
 (**'CLAW'**, 0, **'false'**, **'false'**, **'symmetrical'**, 9, 20, -2, **'Right Shape Value: symmetrical'**),  
 (**'CLAW'**, 0, **'false'**, **'false'**, **'both'**, 9, 20, -2, **'Right Shape Value: both'**),

Wrong Shape Value

Right Shape Value: Asymmetrical

Right Shape Value: symmetrical

Right Shape Value: both

Excpted: when right value is assigned results will come back

When worng value is assgined nothing will come back , and test will check if the value is ok , the test will also ignore Capitol lettres as bad input and will lower them

**no issues found**

---------------------------------------------------------------------------------

**shape():**

Each test case evaluates the behavior of the system under different conditions of shape and leniency, ensuring that the mouse shape is within the set criterias regradless of other params such as leniency or grip type

params = [  
 (**'CLAW'**, 0, **'false'**, **'false'**, **'symmetrical'**, 9, 20, -1, **'symmetrical, leniency set to 0'**),  
 (**'FINGERTIP'**, 2, **'false'**, **'false'**, **'symmetrical'**, 9, 20, -1, **'symmetrical, leniency set to 2'**),  
 (**'CLAW'**, 3, **'true'**, **'false'**, **'both'**, 9, 20, -1, **'both, leniency set to 3'**),  
 (**'PALM'**, 4, **'true'**, **'false'**, **'asymmetrical'**, 9, 20, -1, **'asymmetrical, leniency set to 4'**),

**no issues found**

---------------------------------------------------------------------------------

**test\_performance()**

The test evaluates the system's behavior under various configurations by making three requests per test case.

Each request's performance is measured by recording start and end times, calculating elapsed time, and printing it.

Additionally, it calculates the average time per test case and flags "Bad" if the percentage difference between the max and min times exceeds 70%.

params = [  
 (**'CLAW'**, 0, **'false'**, **'false'**, **'symmetrical'**, 9, 20, -1),  
 (**'FINGERTIP'**, 2, **'false'**, **'false'**, **'symmetrical'**, 9, 20, -1),  
 (**'CLAW'**, 3, **'true'**, **'false'**, **'both'**, 9, 20, -1),  
 (**'PALM'**, 4, **'true'**, **'false'**, **'asymmetrical'**, 9, 20, -1),

**no issues found**