Automated Test: (pseudocode&JS) *GET /v4/catalog/artists/* API route

Test case1: ***Exact name match*** -> Send a GET request with the "name_exact" parameter containing an artist's name

- Description: Filter artists by an exact name match.
- Input: name_exact=Kobe
- Expected Result: HTTP 200 OK, results contain artists with the exact name "Kobe".

```
//1. Define test data
base_url = "https://api.beatportstage.com/v4/catalog/artists/"
artist_name = "Kobe"
//2.Construct a GET request URL by appending the name_exact parameter to the base URL:
url = base_url + "?name_exact=" + artist_name
//2.1 Send a GET request to the constructed URL
response = send_get_request(url)
//3. Validate response. Check the response status code:Expected: 200 OK
if response.status_code != 200:
  report_failure("Unexpected status code: " + str(response.status_code))
  exit
//3.1 Check the response content type: Expected: application/JSON
if response.content_type != "application/json":
  report_failure("Unexpected content type: " + response.content_type)
  exit
//3.2 Parse JSON response
data = parse_json(response.body)
if not data.has_key("Results"): report_failure("Missing 'Results' property in response")
exit
//3.3 Check artist names for artist in data["Results"]:
if artist["name"] != artist_name: report_failure("Artist name mismatch found") exit
//4. Test successful
report_success("Test passed: Get Artists by Exact Name")
```

JS API test pseudocode:

Request:
- Method: GET
- URL: https://api.beatportstage.com/v4/catalog/artists/?name_exact=Kobe

Test:

```
pm.test("Status code is 200", function () {
        pm.response.to.have.status(200);
});
pm.test("Results contain exact name match", function () {
        const jsonData = pm.response.json();
        pm.expect(jsonData.results).to.be.an('array');
        pm.expect(jsonData.results.length).to.be.greaterThan(0);
```

```
    pm.expect(jsonData.results[0].name).to.equal('Kobe');
});
```

Test case 2(negative): *Invalid name filter*

- Description: Filter artists by an invalid name.
- Expected Result: HTTP 400 Bad Request, an error message indicating an invalid name parameter.

```
//1. Define test data
base_url = "https://api.beatportstage.com/v4/catalog/artists/"
invalid_name_filter = "!@#$%^&*"
//2.Construct a GET request URL
url = base_url + "?name=" + invalid_name_filter
//2.1 Send a GET request to the constructed URL
response = send_get_request(url)
//3. Validate response.
if response.status_code not in (400, /* Other expected error codes */):
    report_failure("Unexpected status code for invalid name filter: " +
str(response.status_code))
exit
//3.1 Optional: Check for the error message in the response body (if applicable)
//4. Test successful
report_success("Test passed: Invalid Name Filter")
```

JS API test pseudocode:

1. Status code check: Ensures that the response status code is 400, indicating a bad request due to the invalid parameter.
2. Error message check: Verifies that the response body contains an error property with a string value, which typically signifies that the server recognized the invalid parameter and returned an appropriate error message.

Request:
- Method: GET
- URL: https://api.beatportstage.com/v4/catalog/artists/?name_invalid=InvalidName

```
pm.test("Status code is 400", function () {
    pm.response.to.have.status(400);
});
pm.test("Response contains error message", function () {
    const jsonData = pm.response.json();
    pm.expect(jsonData).to.have.property('error');
    pm.expect(jsonData.error).to.be.a('string');
});
```