**3. API Testing**

**Api documentation:** https://www.videogamedb.uk/swagger-ui/index.html#/

**Write testcases to validate APIs in this document.**

The different test cases that can be conducted on the given document can be represented by CRUD function. The test case can be listed as below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | API Endpoint | Action | Status Code | Expected Response |
| 1 | /api/games/ | Create a new game | 201 | A new game id is created. |
| 2 | /api/games/{id} | Retrieve the game id | 200 | The game id should be shown. |
| 3 | Update the game id | The game id should be updated. |
| 4 | Verify the updated game id | The updated details should be shown. |
| 5 | Delete the game id | 204 | The game should be deleted. |
| 6 | Attempt to get the deleted id | 404 | Error message should be shown. |

**Write test scenario with combination of multiple API (Get, Post, Put, Delete)**

The given document represents the end points to manage video game data. They contain the information of game, developers, publishers, genre and platforms.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Method | API Endpoint | Action | Status Code | Expected Response |
| 1 | GET | /api/games | Retrieve all games | 200 | Should show array of game responses |
| 2 | /api/games/{id} | Retrieve a game with a valid ID | 200 | Should show a game object with {id} |
| 3 | /api/games/85217 | Retrieve a game with invalid id | 404 | Error message should be displayed |
| 4 | POST | /api/games | Add new game with valid data | 201 | A game object is created |
| 5 | /api/games | Add new game with invalid/missing data | 400 | Error message should be displayed |
| 6 | PUT | /api/games/{id} | Update a game with valid id | 200 | The game object is updated |
| 7 | /api/games/13a | Update a game with invalid/mismatched id | 400 | Error message should be displayed |
| 8 | DELETE | /api/games/{id} | Delete a game with valid id | 204 | The game should be deleted |
| 9 | /api/games/85217 | Delete a game with invalid id | 404 | Error message should be displayed |

**Verify Schema of response**

For Schema Validation, let us investigate the CRUD function.

Here, the /api/videogame GET method has the following JSON.

|  |
| --- |
| [  {  "category": "string",  "id": 0,  "name": "string",  "rating": "string",  "releaseDate": "string",  "reviewScore": 0  }  ] |

To check for Schema Validation, I am using the POST Method and the url: /api/authenticate. Here, the json schema is given as:

{“token”: “string”}

This is the response body. The status code for positive response is 200 OK, while for the negative response is 400, Invalid username and password. In the request body, we have provided the given information,

|  |  |
| --- | --- |
| {  “username”: “admin”,  “password”: “admin”  } |  |
| {  “username”: admin,  “password”: “admin”  } |  |

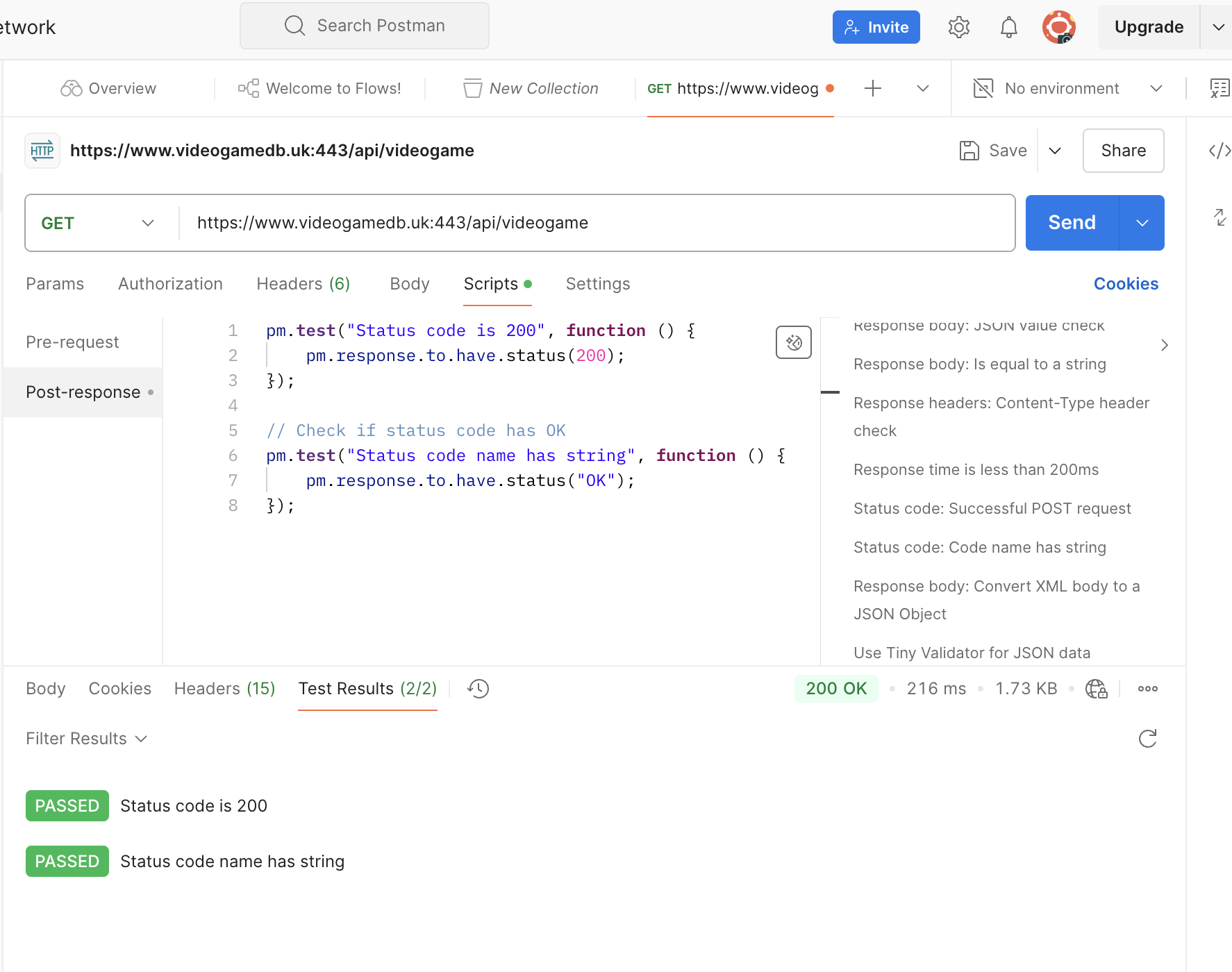
Hence, we can confirm schema validation.

**Using postman implement your testcases and add test scripts**

The postman scripts and snippets of the testcases listed above can be given as follows.

1. Positive Test- Retrieve all the games

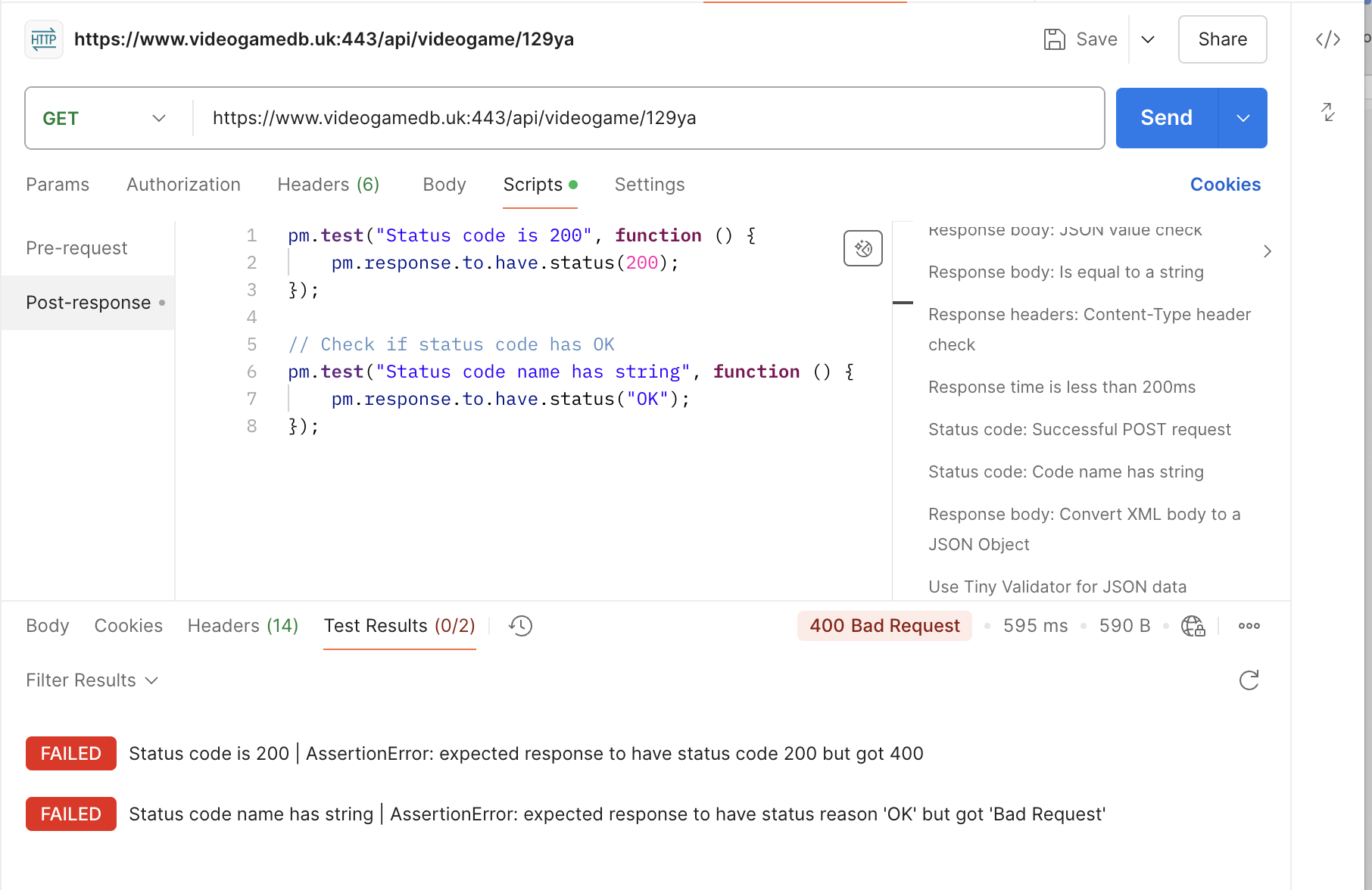
*(If the test is successful, status code 200 should be shown.)*



The given snippet is for the API testing for listing all the videogames. As we can see that the test has a positive response, given we can have a list of all the videogames.

1. Negative Test – Retrieve game with invalid id

*(Error message should be displayed with status code 400- Bad Request)*

**