Homework 06:

/*

Given

https://reqres.in/api/unknown/3

When

User send a GET request to the URL

Then

HTTP Status Code should be 200

And

Response content type is "application/json; charset=utf-8"

And

Response body should be like;(Soft Assertion)

{

"data": {

"id": 3,

"name": "true red",

"year": 2002,

"color": "#BF1932",

"pantone_value": "19-1664"

},

"support": {

"url": "https://reqres.in/#support-heading",

"text": "To keep ReqRes free, contributions towards server costs are appreciated!"

}

}

*/

For "Homework 06," which tasks you with verifying the response from a GET request to an API, you'll be testing for a 200 OK status, appropriate content type, and specific response body content. You can achieve this using Postman for interactive testing or by scripting a test using JavaScript with Node.js and the Axios library. Each approach provides a structured method to confirm that the response adheres to the outlined specifications.

Using Postman for Interactive API Testing

1. **Open Postman**:
    - Launch the Postman application.
2. **Configure the GET Request**:
    - Set the method to GET.
    - Enter the URL: **https://reqres.in/api/unknown/3**.
3. **Implement Tests**:
    - Navigate to the "Tests" tab in the request setup.
    - Insert the following JavaScript code to assess the response against the specified criteria:

```
// Check for the correct HTTP status code pm.test("HTTP Status Code is 200", function () {
pm.response.to.have.status(200); }); // Confirm the content type pm.test("Response content type is
application/json; charset=utf-8", function () { pm.response.to.have.header("Content-Type",
"application/json; charset=utf-8"); }); // Validate the response body with soft assertions let
actualResponseBody = pm.response.json(); pm.test("Response body validation", function () {
pm.expect(actualResponseBody.data.id).to.eql(3);
pm.expect(actualResponseBody.data.name).to.eql("true red");
pm.expect(actualResponseBody.data.year).to.eql(2002);
pm.expect(actualResponseBody.data.color).to.eql("#BF1932");
pm.expect(actualResponseBody.data.pantone_value).to.eql("19-1664");
pm.expect(actualResponseBody.support.url).to.eql("https://reqres.in/#support-heading");
pm.expect(actualResponseBody.support.text).to.eql("To keep ReqRes free, contributions towards server
costs are appreciated!"); });
```

4. **Send the Request and Review**:
    - Execute the request and observe the results in the "Test Results" tab to confirm that all conditions are satisfied.

Using JavaScript and Axios for Programmatic API Testing

1. **Prepare Your Environment**:
    - Ensure Node.js is installed.

- Create a directory, initialize a Node.js project, and install Axios:

mkdir my-api-test cd my-api-test npm init -y npm install axios

2. **Write the Test Script**:
    - Create a file named **test.js**.
    - Add the following JavaScript code to send a GET request and validate the response:

const axios = require('axios'); axios.get('https://reqres.in/api/unknown/3') .then(response => { console.log("HTTP Status Code check:", response.status === 200); // Should be true console.log("Content-Type check:", response.headers['content-type'].includes("application/json; charset=utf-8")); // Should be true // Detailed checks for response body content console.log("Data ID check:", response.data.data.id === 3); console.log("Data Name check:", response.data.data.name === "true red"); console.log("Data Year check:", response.data.data.year === 2002); console.log("Data Color check:", response.data.data.color === "#BF1932"); console.log("Pantone Value check:", response.data.data.pantone_value === "19-1664"); console.log("Support URL check:", response.data.support.url === "https://reqres.in/#support-heading"); console.log("Support Text check:", response.data.support.text === "To keep ReqRes free, contributions towards server costs are appreciated!"); }) .catch(error => { console.error('Error during API call', error); });

3. **Execute the Test**:
    - Run the script by typing:

node test.js

Both methods—using Postman for an interactive testing experience or JavaScript for automated testing—provide robust mechanisms to ensure that the API behaves as expected according to the detailed requirements specified in your homework assignment.