Homework 05:

/*

Given

https://reqres.in/api/users/23

When

User send a GET Request to the url

Then

HTTP Status code should be 404

And

Status Line should be HTTP/1.1 404 Not Found

And

Server is "cloudflare"

And

Response body should be empty

*/

For "Homework 05," which involves sending a GET request to an API endpoint expected to return a 404 Not Found error, you can utilize JavaScript with Node.js and the Axios library to perform this testing. This approach will check for the appropriate HTTP status, status line, server type, and verify that the response body is indeed empty, as per the assignment's requirements.

Using JavaScript and Axios for Programmatic API Testing

Step 1: Prepare Your Environment

- **Ensure Node.js is Installed**: Verify that Node.js is available on your system. If not, you will need to install it.
- **Set Up Your Project**:

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

Step 2: Write the Test Script

- **Create a Test Script**: Generate a new file named **test.js**.

- **Script Content**: Implement the following JavaScript code to send a GET request and validate the response according to your homework's specific conditions:

const axios = require('axios'); axios.get('https://reqres.in/api/users/23') .then(response => { // If the request does not fail, this block unexpectedly executes console.log("Request unexpectedly succeeded."); }) .catch(error => { // Expected error handling block console.log("HTTP Status Code check:", error.response.status === 404); // Check for status code 404 console.log("Status Line check:", error.response.statusText === "Not Found"); // Check for correct status text console.log("Server check:", error.response.headers.server === "cloudflare"); // Check server header console.log("Response body is empty:", JSON.stringify(error.response.data) === "{}"); // Check if the response body is empty });

Step 3: Execute the Test

- **Run the Script**:

node test.js

Key Academic Insights

- **HTTP Status Code Validation**: The script ensures that the server returns a 404 status code, which is appropriate for a resource that cannot be found.
- **Response Content Checking**: It verifies that the content type and body are as expected for a 404 error — typically, the body should be empty.
- **Server Header Validation**: Confirming the server type as "cloudflare" checks for consistency in the API's infrastructure response, which can be critical for debugging and security assessments.
- **Error Handling**: This script is specifically designed to handle expected errors gracefully, which is a crucial part of robust API testing and ensures that the API behaves correctly under error conditions.

This approach is well-suited for backend testing and can be integrated into automated test suites or continuous integration systems to ensure consistent API behavior.