Homework 08:

```
/*
    Given
      1) https://reqres.in/api/users
      2) {
         "name": "morpheus",
         "job": "leader"
         }
    When
      I send POST Request to the Url
    Then
      Status code is 201
      And response body should be like {
                                  "name": "morpheus",
                                  "job": "leader",
                                  "id": "496",
                                  "createdAt": "2022-10-04T15:18:56.372Z"
                                 }
 */
```

For "Homework 08," which involves sending a POST request to an API endpoint and validating the response to ensure it matches expected status codes and data, you can efficiently execute this using JavaScript with Node.js and the Axios library. This method allows for programmatically posting data, checking response status, and ensuring the response body content matches specified criteria.

Using JavaScript and Axios for Programatic API Testing

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

- 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 POST request and validate the response according to the specifications provided:

const axios = require('axios'); const userData = { "name": "morpheus", "job": "leader" }; axios.post('https://reqres.in/api/users', userData) .then(response => { console.log("HTTP Status Code check:", response.status === 201); // Status code 201 check console.log("Response body:", response.data); // Assertions to validate response body content console.assert(response.data.name === "morpheus", `Expected name to be 'morpheus', but got '${response.data.name}'`); console.assert(response.data.job === "leader", `Expected job to be 'leader', but got '${response.data.job}'`); console.assert(response.data.hasOwnProperty('id'), "Expected 'id' field in the response"); console.assert(response.data.hasOwnProperty('createdAt'), "Expected 'createdAt' field in the response"); // Printing the 'id' and 'createdAt' to ensure they are present and correct format console.log("ID:", response.data.id); console.log("Created At:", response.data.createdAt); }) .catch(error => { console.error('Error during API call', error); });

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

node test.js

This script:

- Sends a POST request with the specified user data (**name** and **job**).
- Validates that the response has a status code of 201.
- Checks that the response body contains the fields **name**, **job**, **id**, and **createdAt**, with correct values for **name** and **job**.
- Asserts the presence of **id** and **createdAt**, which are typically auto-generated by the server.

Key Academic Points

- **HTTP Status Code**: The test checks for a 201 status code, indicating that a resource has been successfully created.

- **Response Validation**: The test asserts that the response body matches the expected structure and content, confirming that the server handles data correctly and responds with dynamically generated fields.
- **Automation**: This script is suitable for inclusion in a continuous integration pipeline to ensure that backend services consistently handle post requests correctly.

This approach to API testing is highly effective for backend development and testing practices, providing a robust method to ensure APIs behave as expected under specified operational conditions.