***API*** ***Assignment Questions***

1. **Explain the following:**
2. PUT and PATCH methods

Answer: **PUT** and **PATCH** are both HTTP methods used to **update resources** on a server, but they have different use cases.

| **Feature** | **PUT** | **PATCH** |
| --- | --- | --- |
| Type of update | **Full update** of a resource | **Partial update** of a resource |
| Behavior | Replaces the entire existing resource | Updates only the fields provided in the request |
| Idempotency | **Yes** – Sending it multiple times gives the same result | **Yes**, but depends on implementation |
| Example | Updating all user details | Updating only the email or phone number |

1. Headers and Cookies

Answer: **Headers**

Headers are **key-value pairs** sent in the request or response that provide **metadata** about the request or response.

**Common types of headers:**

| **Header Name** | **Purpose** |
| --- | --- |
| Content-Type | Tells the server the format of the request body (e.g., application/json) |
| Authorization | Used to pass credentials like tokens |
| Accept | Tells the server what kind of response formats are accepted (e.g., application/json) |

**Example:**

http

CopyEdit

GET /users HTTP/1.1

Host: example.com

Content-Type: application/json

Authorization: Bearer abc123token

**🔹 Cookies**

Cookies are small pieces of data stored on the **client side** and sent back to the **server** with each request.

They are often used for:

* **Maintaining sessions** (e.g., keeping users logged in)
* **Tracking user behavior**
* **Personalizing content**

**Example of a Cookie header:**

http

CopyEdit

Cookie: sessionId=abc123xyz

1. Endpoint and Base URL

Answer: An **endpoint** is the **specific path** added to the base URL to access a particular resource or function of the API.

**Example:**

If you want to get a list of users:

arduino

https://reqres.in/api/users

* https://reqres.in/api → Base URL
* /users → Endpoint

**Base URL**

The **Base URL** is the **root address** of the API you're accessing. It stays **constant** for all requests made to the API.

**Example:**

For the ReqRes API:

arduino

https://reqres.in/api

Here, https://reqres.in/api is the base URL.

**Together:**

**Base URL + Endpoint = Complete URL for a specific request**

1. Query Parameters and Path Parameters

Answer: **Query Parameters**

* Appended to the **end of the URL** after a ?
* Used to **filter or customize** the response
* Format: ?key=value&key2=value2

**Example:**

nginx

GET https://reqres.in/api/users?page=2

* Here, page=2 is a **query parameter**, used to request the second page of users.

**Path Parameters**

* Part of the **URL path**
* Used to identify a **specific resource**
* Typically written in curly braces when defining routes

**Example:**

bash

GET /users/{id}

Requesting user with ID 5:

ruby

CopyEdit

GET https://reqres.in/api/users/5

* Here, 5 is a **path parameter** representing a specific user.

1. What are error codes? Explain all series of error codes.

Answer: **What are Error Codes?**

**Error codes** are part of the HTTP status codes, specifically in the **400 and 500 series**, that inform you something went wrong.

**🔹 All HTTP Status Code Series:**

| **Code Series** | **Meaning** | **Example Codes** | **Description** |
| --- | --- | --- | --- |
| **1xx** | Informational | 100, 101 | Request received, continuing process |
| **2xx** | Success | 200, 201 | Request was successful |
| **3xx** | Redirection | 301, 302 | Further action is needed (e.g., URL has moved) |
| **4xx** | Client Error | 400, 401, 403, 404 | The request has a mistake from the client’s side |
| **5xx** | Server Error | 500, 502, 503 | Server failed to process a valid request |

**🔹 Common Error Codes:**

| **Code** | **Meaning** | **Description** |
| --- | --- | --- |
| 400 | Bad Request | The server cannot understand the request due to invalid syntax |
| 401 | Unauthorized | Authentication is needed or has failed |
| 403 | Forbidden | Server understood but refuses to authorize |
| 404 | Not Found | The resource was not found |
| 405 | Method Not Allowed | HTTP method is not supported for this resource |
| 500 | Internal Server Error | Generic server-side error |
| 502 | Bad Gateway | Invalid response from the upstream server |
| 503 | Service Unavailable | Server is down or overloaded |

1. Create a new collection in Postman named "Sample APIs." Use https://reqres.in/ and implement GET, POST, PUT, PATCH, and DELETE operations.

Answer: https://api.postman.com/collections/46085162-0d0f8679-46bc-4c18-a723-6e21e8751c4a?access\_key=PMAT-01JYE5E3DNCNTB4W1B8SB4ZSSR

1. What is the difference between given(), when(), and then() in Rest Assured?

Answer:

| **Method** | **Purpose** | **Example Role** |
| --- | --- | --- |
| given() | **Set up preconditions** | Headers, parameters, request body |
| when() | **Trigger the request** | HTTP method like GET, POST, etc. |
| then() | **Verify the response** | Status code, response body, assertions |

**Example:**

java

given()

.baseUri("https://reqres.in")

.header("Content-Type", "application/json")

.body("{ \"name\": \"Shtakshi\", \"job\": \"Tester\" }")

.when()

.post("/api/users")

.then()

.statusCode(201)

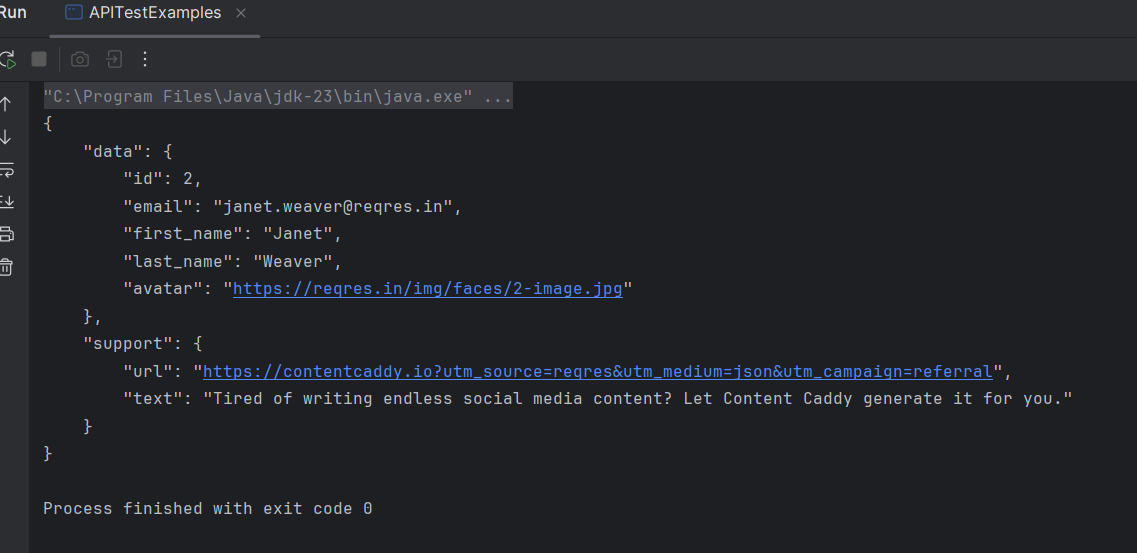
.log().body();

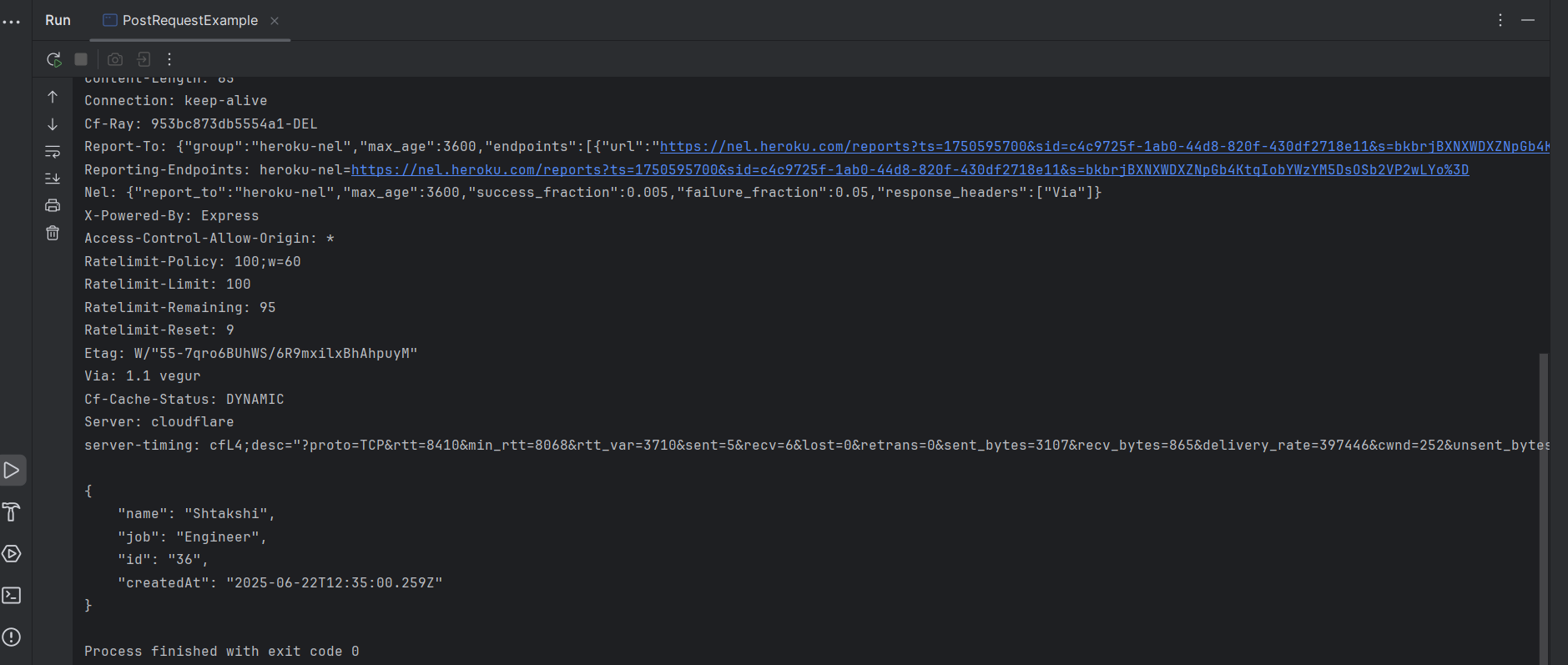
**In Simple Words:**

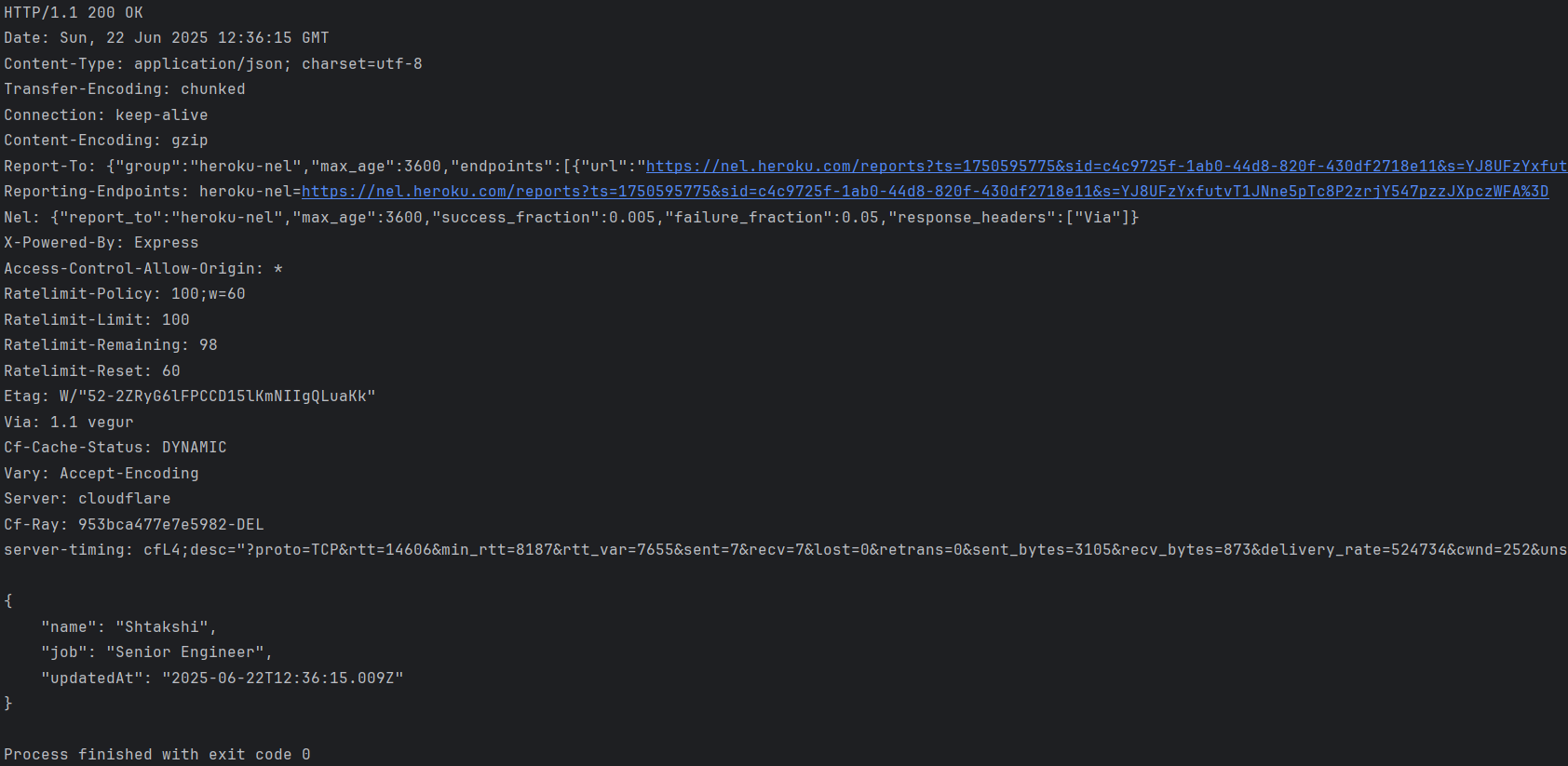
* given() → **What you send** (setup request)
* when() → **What you do** (send request)
* then() → **What you expect** (assert/validate)

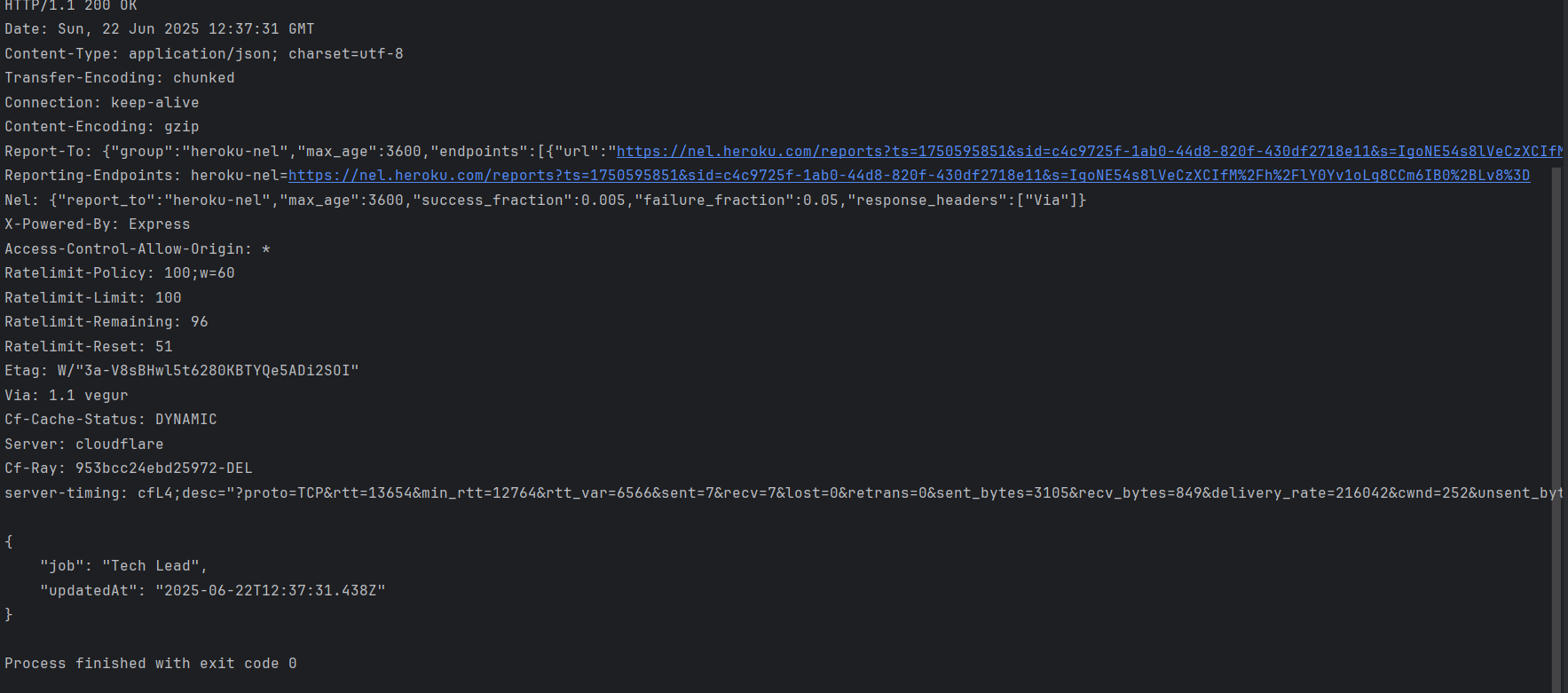
1. **Perform the operations with Rest Assured (use** [**https://reqres.in/**](https://reqres.in/)**):**
2. Create GET, POST, PUT, PATCH, and DELETE

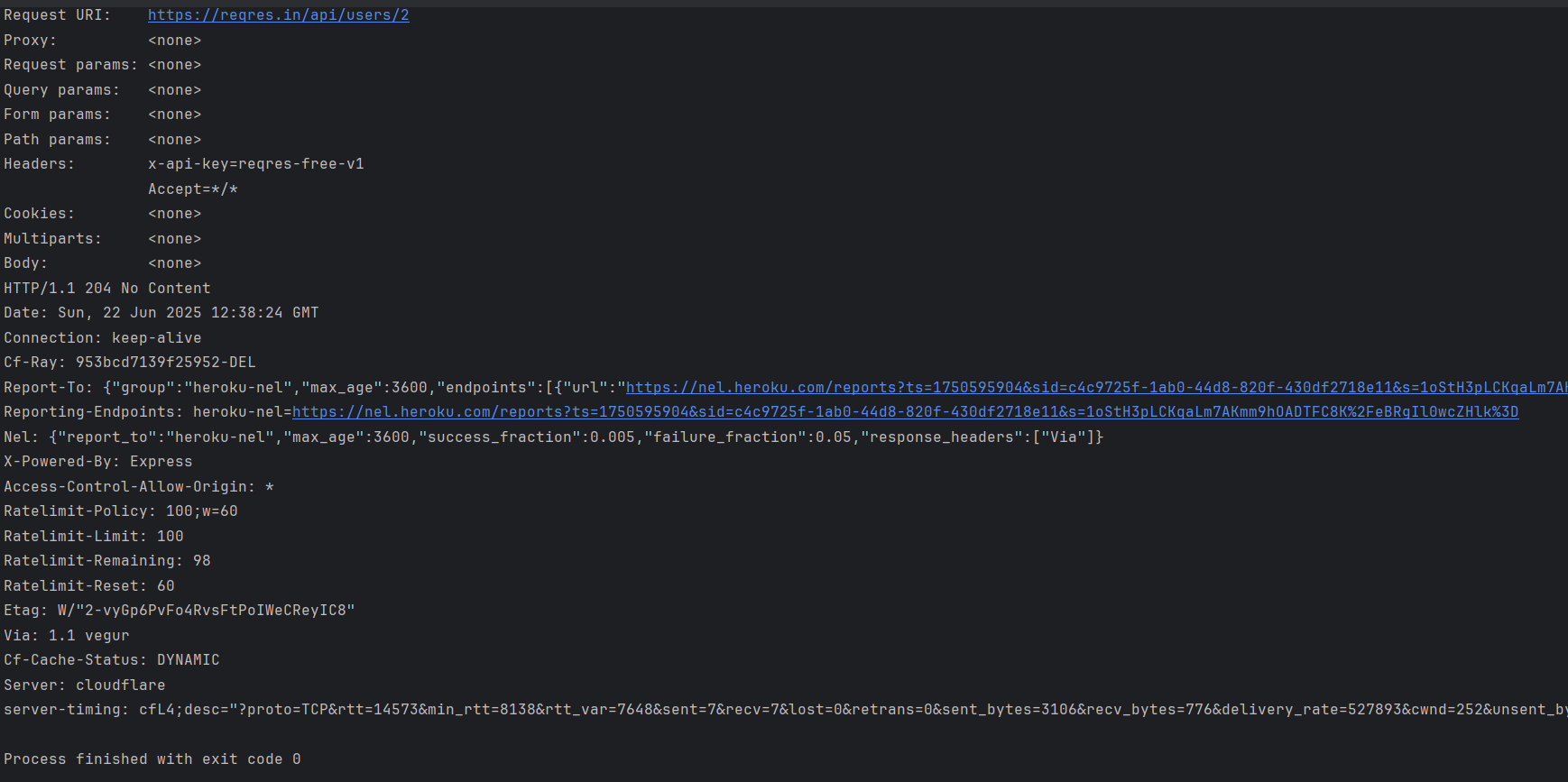
Answer:

GET: 

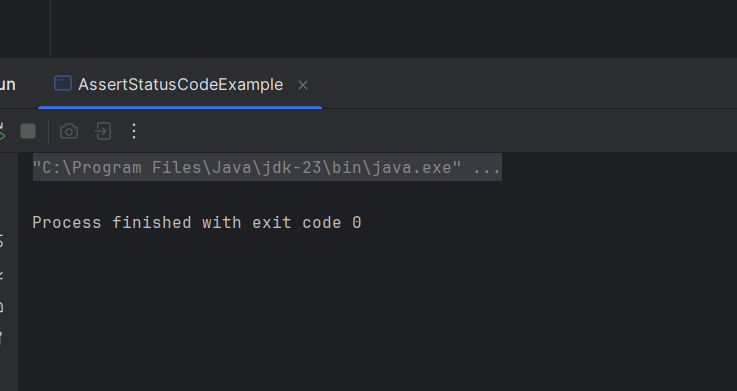
POST: 

PUT: 

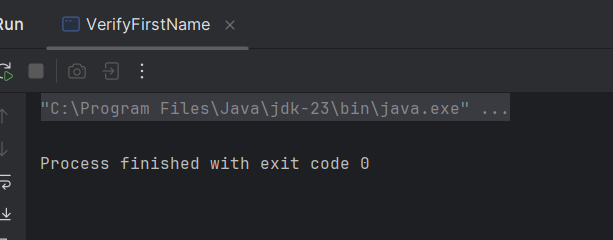
PATCH: 

DELETE: 

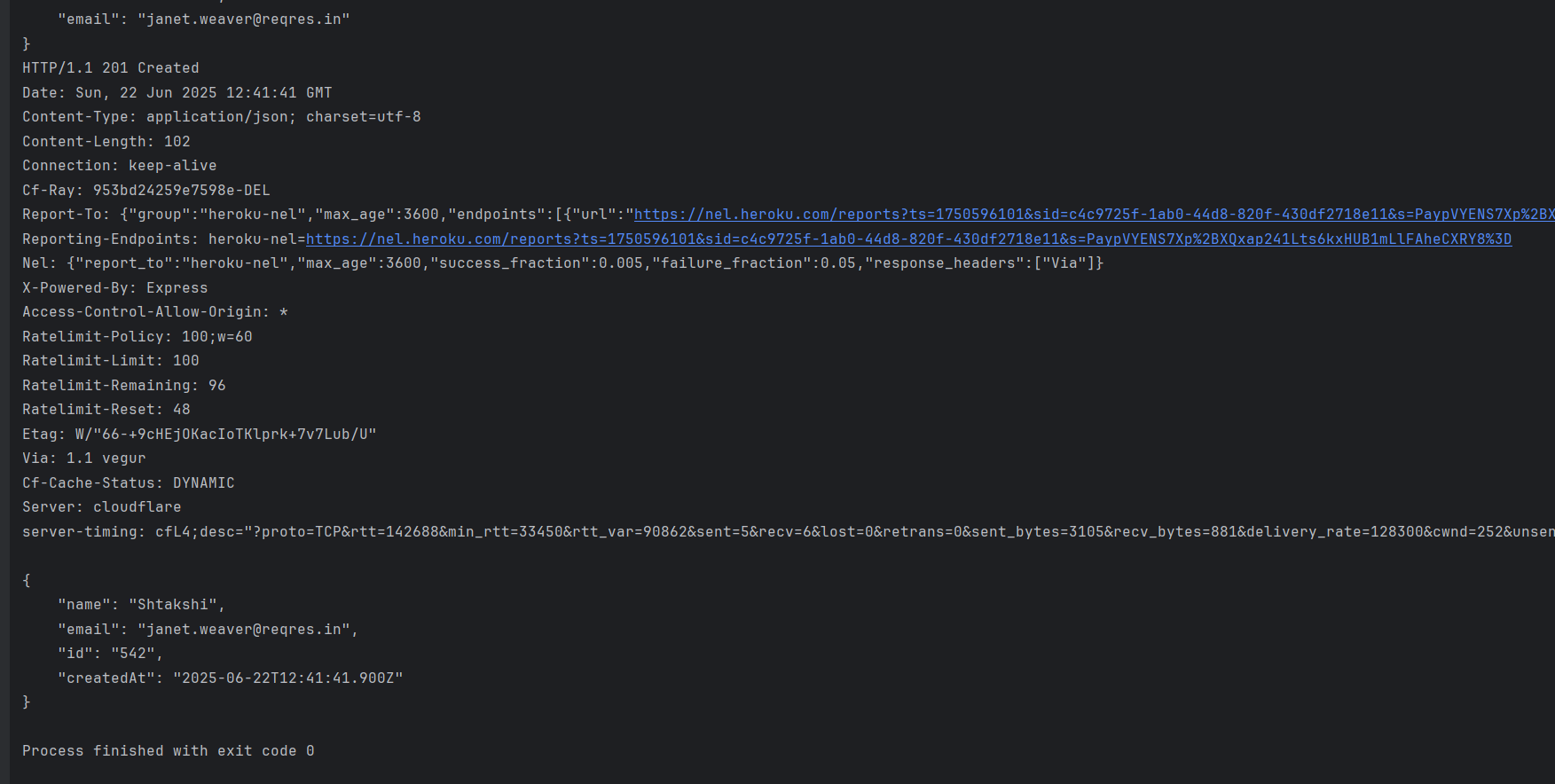
1. Assert for 200/201 response

Answer: 

1. Verify “first\_name” key value from GET response

Answer: 

1. Extract “email” from GET request and add it to the subsequent POST request payload and hit the request

Answer: 

1. Write a function to verify presence of Message in response body

Answer: 