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

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

**-> PUT:**

The PUT method is used to update or replace a resource on the server with the provided data. When you send a PUT request, you are usually sending the entire new representation of the resource (i.e., replacing the existing one). If the resource doesn’t exist, a PUT request can create it.

Example: Updating a user's profile where you provide all the data (e.g., name, email, etc.), and it will completely replace the existing data.

Idempotent: Calling PUT multiple times with the same data will result in the same state.

**-> The PATCH** method is used to update a resource partially. It sends only the fields that need to be modified, rather than replacing the entire resource. This makes it more efficient in situations where only certain parts of a resource need updating.

Example: Updating just the user's email address, not the entire profile.

Not necessarily idempotent: A PATCH request might cause a different state if called multiple times.

1. Headers and Cookies

**-> Headers**:

HTTP headers are key-value pairs sent between the client and server in an HTTP request or response. They provide additional information about the request or response, such as content type, authorization credentials, etc.

Request Headers: Can include information like Authorization, Content-Type, User-Agent, etc.

Response Headers: Can include information like Content-Length, Cache-Control, Set-Cookie, etc.

**-> Cookies:**

Cookies are small pieces of data stored by the browser and sent with HTTP requests to the server. They can hold stateful information, like session identifiers or user preferences.

Set by the server with the Set-Cookie header in the response, and sent back to the server with the Cookie header in subsequent requests.

Cookies can have attributes like expires, domain, path, and secure

1. Endpoint and Base URL

**-> Base URL:**

The base URL is the root address or domain for the API, from which all API endpoints are built. It typically includes the protocol (https or http), the domain, and sometimes the version of the API.

An endpoint is a specific path or route appended to the base URL to access a specific resource or functionality within an API. Endpoints define the action that will be performed (such as retrieving data or submitting a form).

1. Query Parameters and Path Parameters

**-> Query Parameters:**

Query parameters are key-value pairs appended to the URL after the ? symbol. They are typically used for filtering, sorting, or other modifications of the data being requested. Query parameters are part of the URL but are optional and usually non-hierarchical.

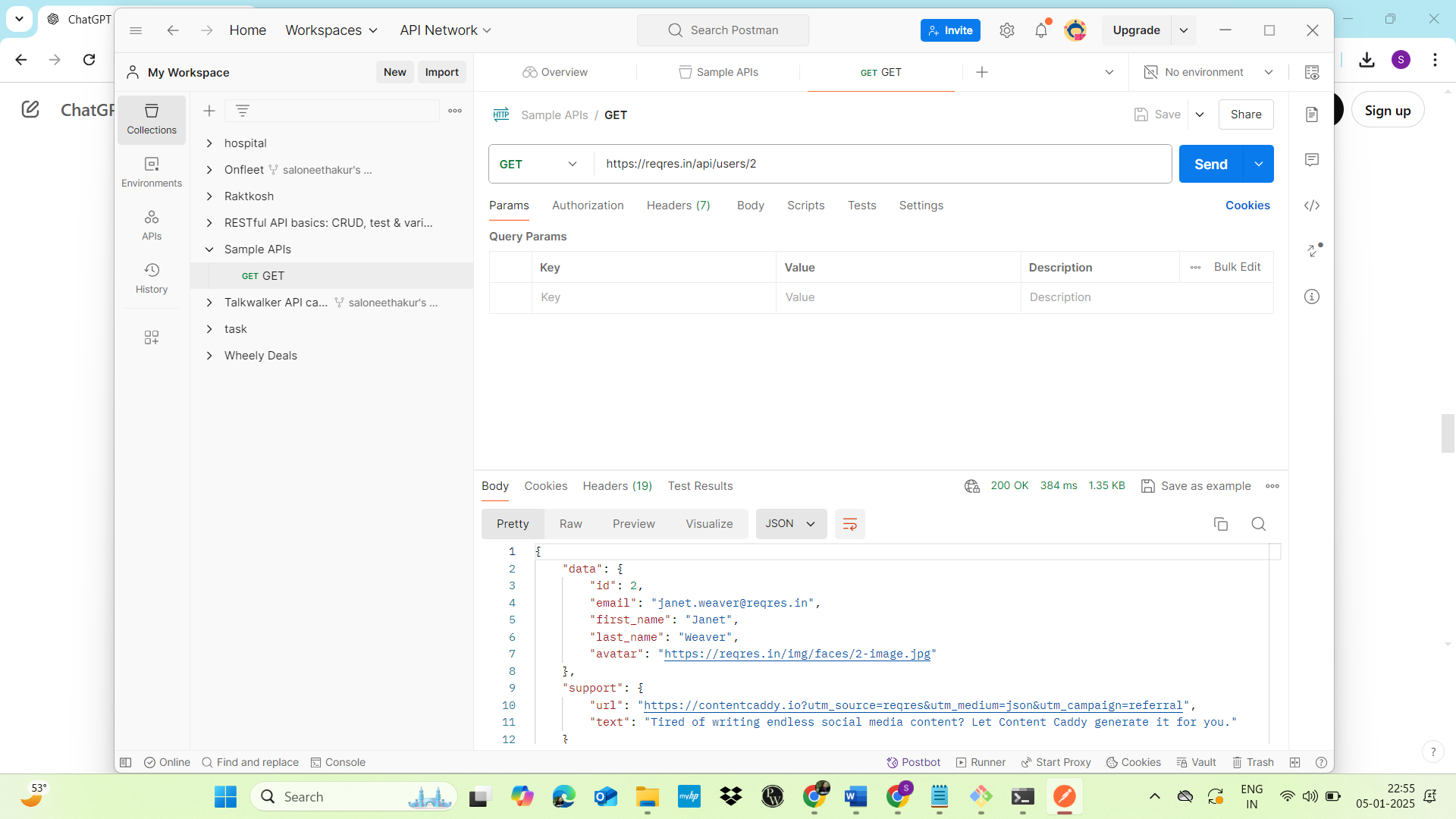
**-> Path Parameters:**

Path parameters are dynamic parts of the URL path. They are used to identify a specific resource or a subset of resources. Path parameters are required and typically represent specific entities or resources.

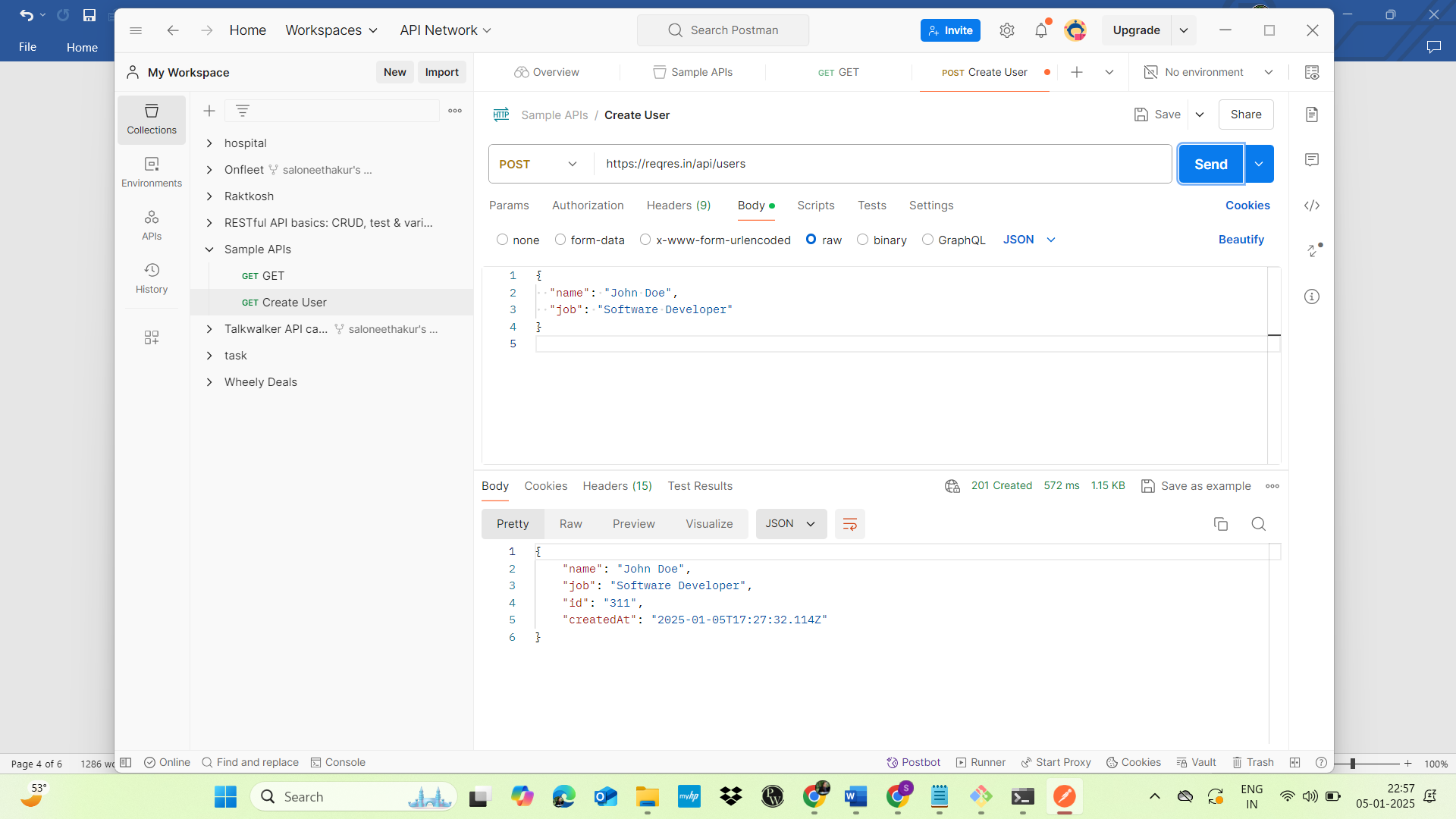
1. What are error codes? Explain all series of error codes.

-> Error codes are part of the HTTP response sent by the server to the client. They indicate the status of the request (whether successful, failed, or redirected). HTTP status codes are categorized into five classes, each representing a different aspect of the response.

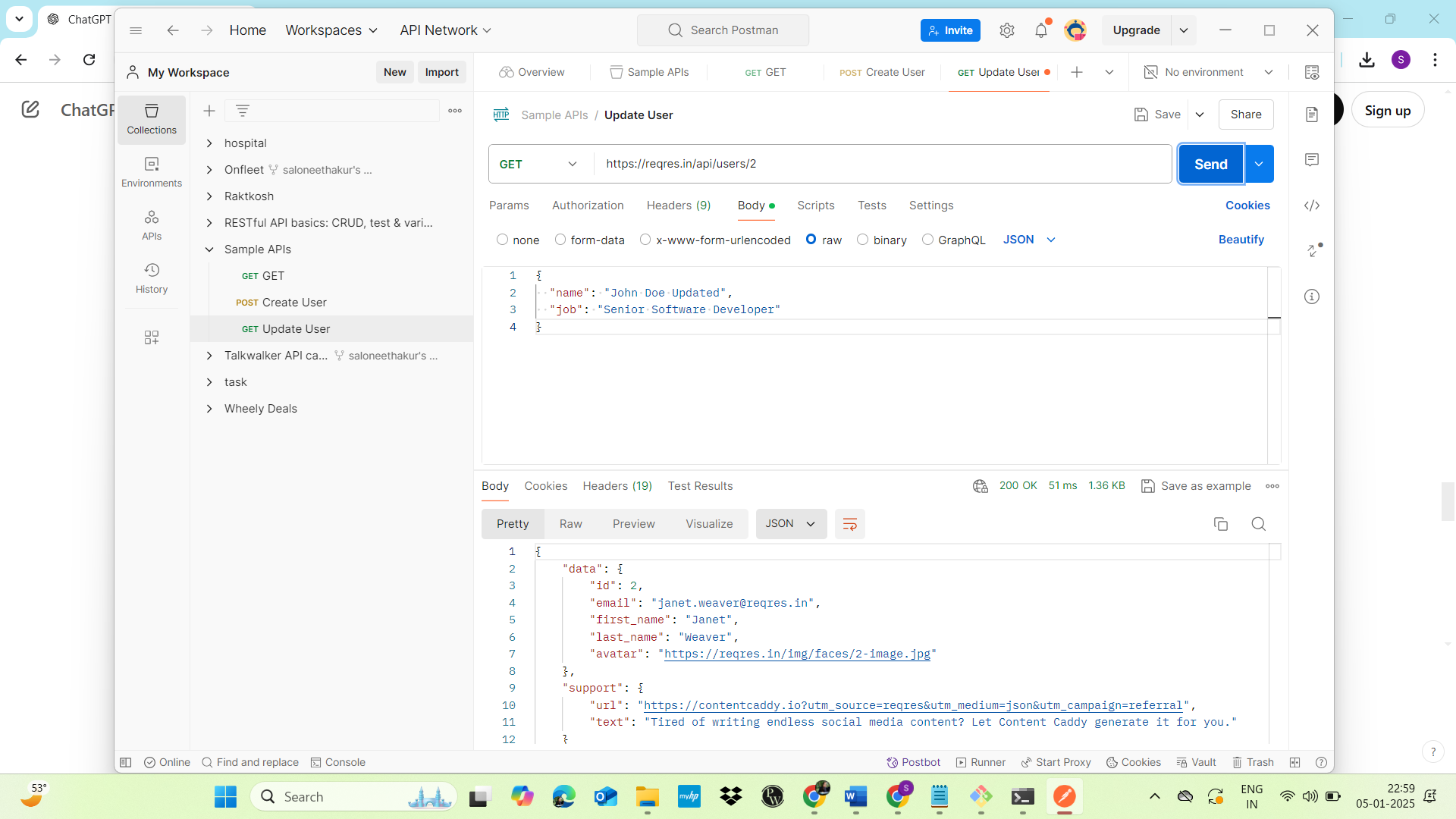
1. **1xx - Informational**  
   These codes indicate that the request is being processed and that the client should wait for further instructions.
   * **100 Continue**: The server has received the request headers, and the client should send the body.
   * **101 Switching Protocols**: The server is switching protocols, as requested by the client.
2. **2xx - Success**  
   These codes indicate that the request was successfully received, understood, and accepted.
   * **200 OK**: The request has succeeded, and the response body contains the result (usually with a GET request).
   * **201 Created**: The request has been fulfilled, and a new resource has been created (usually with a POST request).
   * **204 No Content**: The request has been processed successfully, but there is no content to send in the response.
3. **3xx - Redirection**  
   These codes indicate that the client must take additional actions to complete the request (usually by following a redirect).
   * **301 Moved Permanently**: The resource has been permanently moved to a new URL.
   * **302 Found**: The resource is temporarily located at a different URL.
   * **304 Not Modified**: The resource has not been modified since the last request.
4. **4xx - Client Errors**  
   These codes indicate that the request contains bad syntax or cannot be fulfilled due to issues on the client-side.
   * **400 Bad Request**: The server cannot process the request due to invalid syntax.
   * **401 Unauthorized**: The client must authenticate itself to get the requested response.
   * **403 Forbidden**: The server understands the request but refuses to authorize it.
   * **404 Not Found**: The server cannot find the requested resource.
   * **405 Method Not Allowed**: The method specified in the request is not allowed for the resource.
   * **422 Unprocessable Entity**: The server understands the request, but it cannot process it due to semantic errors (common in REST APIs).
5. **5xx - Server Errors**  
   These codes indicate that the server failed to fulfill a valid request, meaning the issue is on the server-side.
   * **500 Internal Server Error**: The server encountered an unexpected condition that prevented it from fulfilling the request.
   * **502 Bad Gateway**: The server, while acting as a gateway or proxy, received an invalid response from the upstream server.
   * **503 Service Unavailable**: The server is currently unavailable (e.g., due to maintenance or overload).
   * **504 Gateway Timeout**: The server did not receive a timely response from the upstream server while acting as a gateway.
6. Create a new collection in Postman named "Sample APIs." Use https://reqres.in/ and implement GET, POST, PUT, PATCH, and DELETE operations.



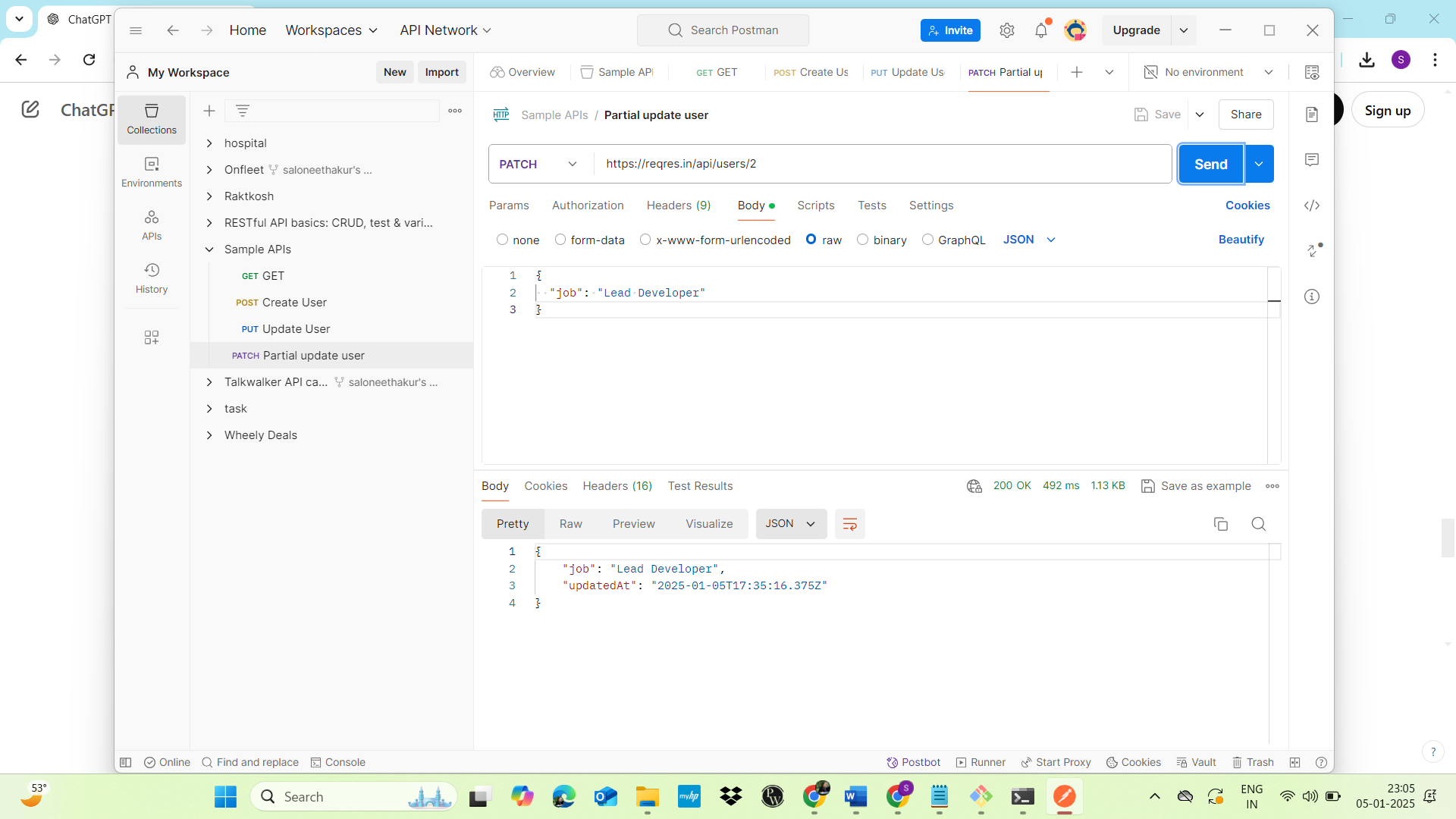
POST:-



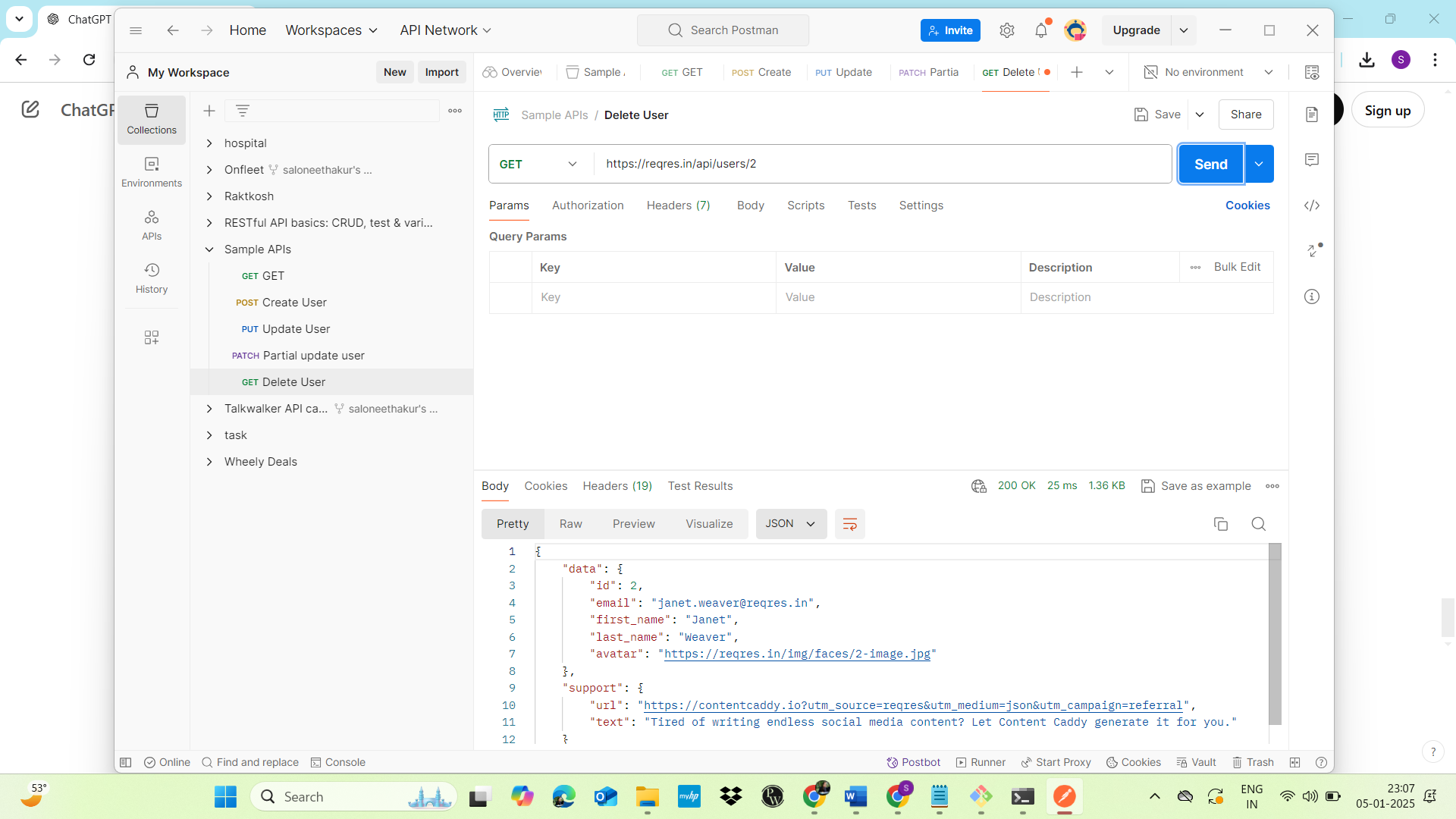
PUT:-



PATCH:-



DELETE:-



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

-> In **Rest Assured**, the given(), when(), and then() methods are used to build and structure API tests in a readable and understandable way. These methods follow the **BDD (Behavior-Driven Development)** style syntax, which makes tests more human-readable. They represent different stages of an API request and the corresponding assertions in the response.

Here's what each of these methods does:

**1. given()**

The given() method is used to **set up the preconditions** or the context for the request. This includes specifying things like request parameters, headers, authentication, and body content. Essentially, it’s where you define the conditions needed to send the HTTP request.

* **Purpose**: Prepare the request and define any setup needed before the actual request is sent.
* **Usage Examples**:
  + Set query parameters (params()).
  + Add request headers (header()).
  + Set the request body (body()).
  + Define authentication (auth()).

### 2. **when**()

The when() method is used to **specify the action or the HTTP request** you want to perform. It’s where you define the type of HTTP method (like GET, POST, PUT, DELETE) and the target endpoint.

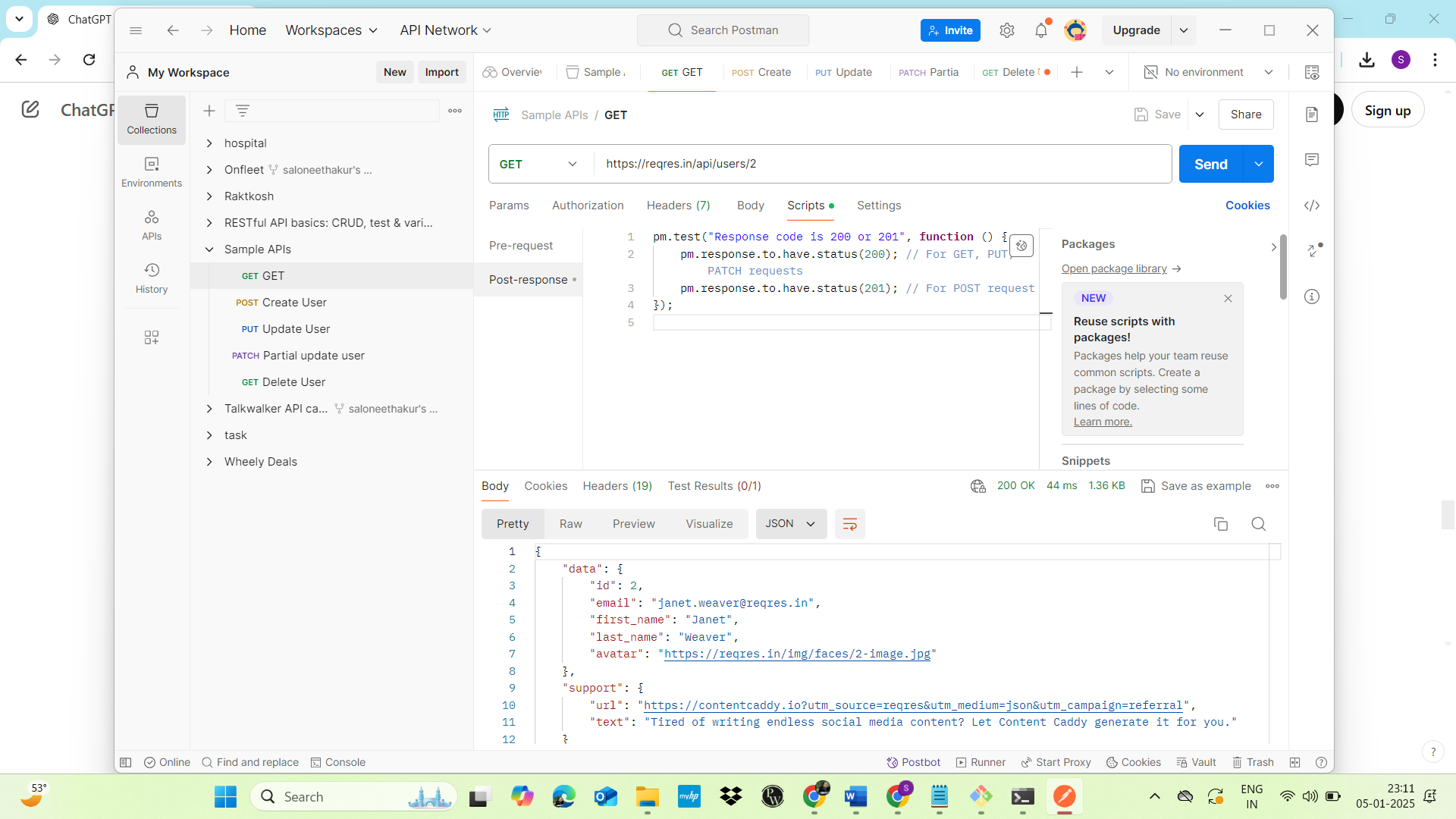
* **Purpose**: Perform the action (send the HTTP request).
* **Usage Examples**:
  + Make a GET, POST, PUT, or DELETE request.

### 3**. then()**

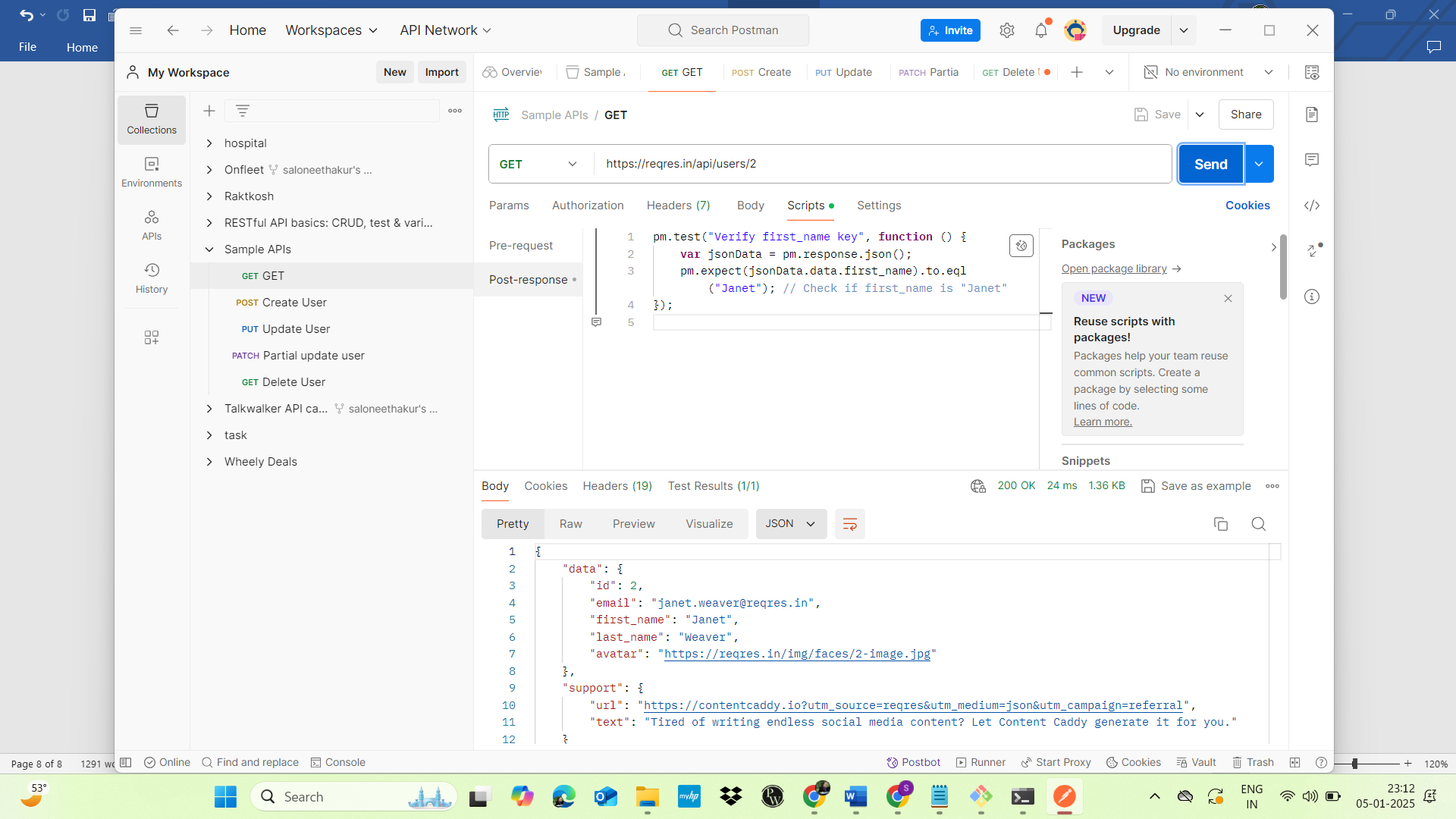
The then() method is used to **define the assertions or verifications** for the response. This is where you validate things like the response status code, headers, body content, and other conditions that should be met after the request is processed.

* **Purpose**: Make assertions on the response returned by the API.
* **Usage Examples**:
  + Check the response status code (statusCode()).
  + Assert the response body (body()).
  + Validate headers (header()).

1. **Perform the operations with Rest Assured (use** [**https://reqres.in/**](https://reqres.in/)**):**
2. Create GET, POST, PUT, PATCH, and DELETE(Already Done)
3. Assert for 200/201 response



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



1. Extract “email” from GET request and add it to the subsequent POST request payload and hit the request
2. Write a function to verify presence of Message in response body

