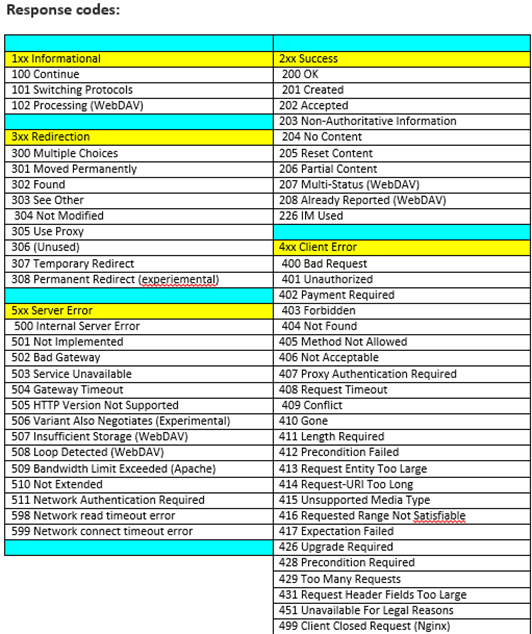
**REST API testing**

**Webpage for Ref :**

**https://restfulapi.net/http-methods/**



* REST stands for Representational State Transfer.
* RESTful web services use the HTTP protocol as a medium of communication between the client and the server.
* RESTful web service usually defines a URI, Uniform Resource Identifier a service, provides resource representation like JSON and a set of HTTP methods.
* HTTP methods Request - GET, PUT, POST, DELETE.
  + GET – Fetch - is only used to request data from a specified resource.
  + POST – CREATE/UPDATE - is used to send data to a server to create/update a resource.
  + PUT – CREATE - replaces all current representations of the target resource with the request payload.
  + DELETE – Delete - removes the specified resource.
  + OPTIONS is used to describe the communication options for the target resource.
  + HEAD - Get status of method availability.
  + Produces - States the HTTP Response generated by web service
  + Consumes - States the HTTP Request type.
* URI’s format is <protocol>://<service-name>/<ResourceType>/<ResourceID>.
* CODES :
  + 200 – Response OK(Successful)
  + 201 – Created when a resource is successfully created using POST or PUT request.
  + 204 - NO CONTENT −when response body is empty. For example, a DELETE request.
  + 400 - Bad request. Required fields missing from input or are malformed.
  + 401 - Unauthorized. Authentication failure; invalid credentials.
  + 402 - Payment required. No active subscription found.
  + 403 – Forbidden status.
  + 404 – Not found status, json contains error or not.
  + 405 - inappropriate for the URL(Condition/parameter is wrong)
  + 409 - CONFLICT − states conflict situation while executing the method
  + 500 - Internal server error. General service failure; retry request.
* Testing REST API
  + Advanced Rest Client
  + SOUP UI
  + Postman-Rest Client
  + Curl in Linux

API URL for testing : <https://reqres.in/>

GET HTTP Method :

Ex : get the single user Data

URL : https://reqres.in/api/users/2

**JSON format:**

{

"data": {

"id": 2,

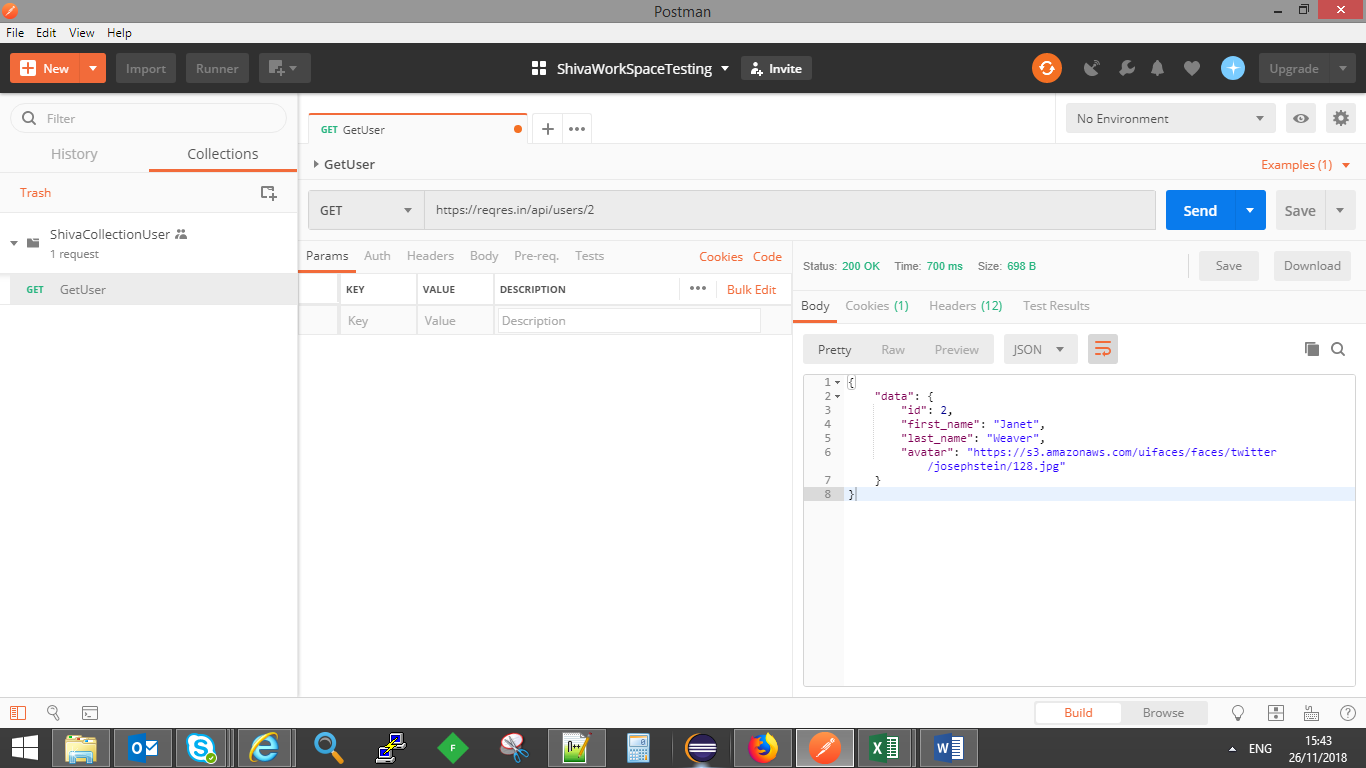
"first\_name": "Janet",

"last\_name": "Weaver",

"avatar": "https://s3.amazonaws.com/uifaces/faces/twitter/josephstein/128.jpg"

}

}



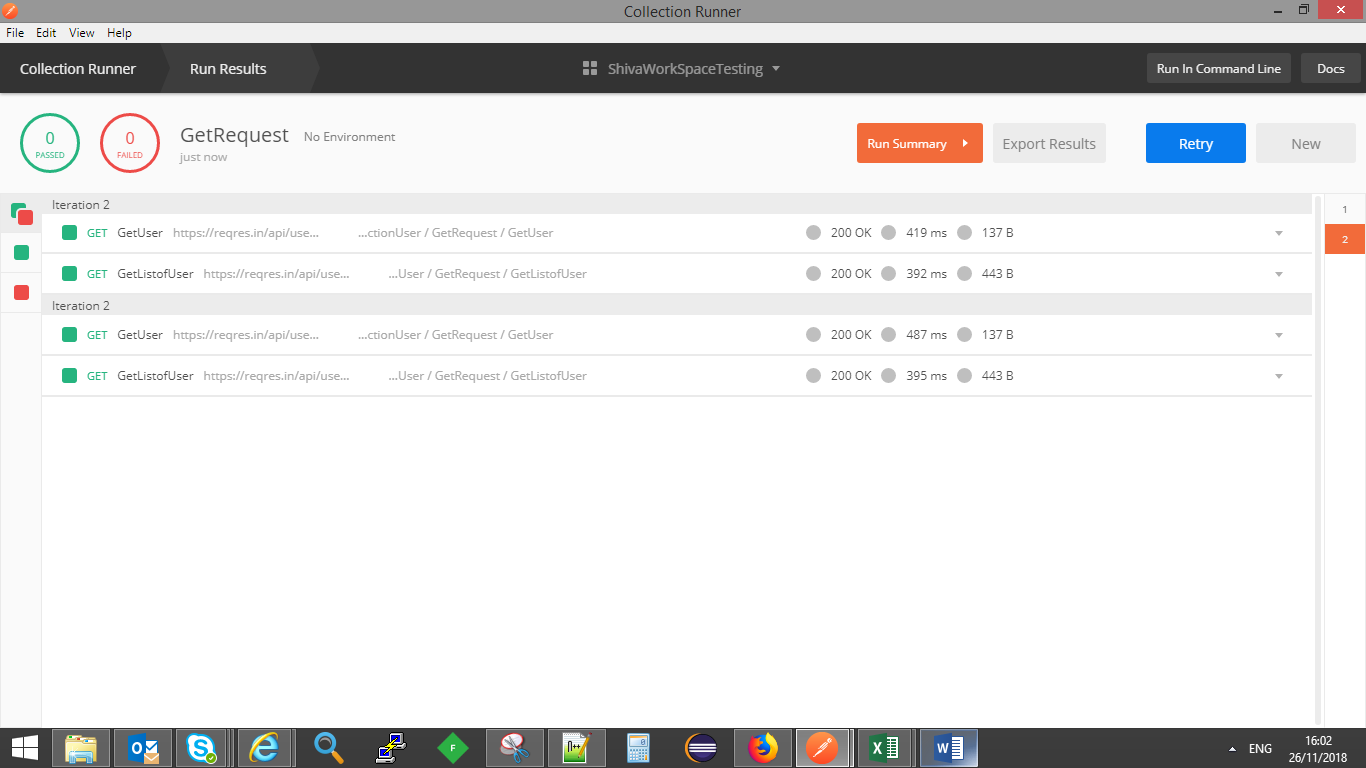
Collection: Group of API requests that can be stored and saved in logical arrangement.

Multiuser:

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

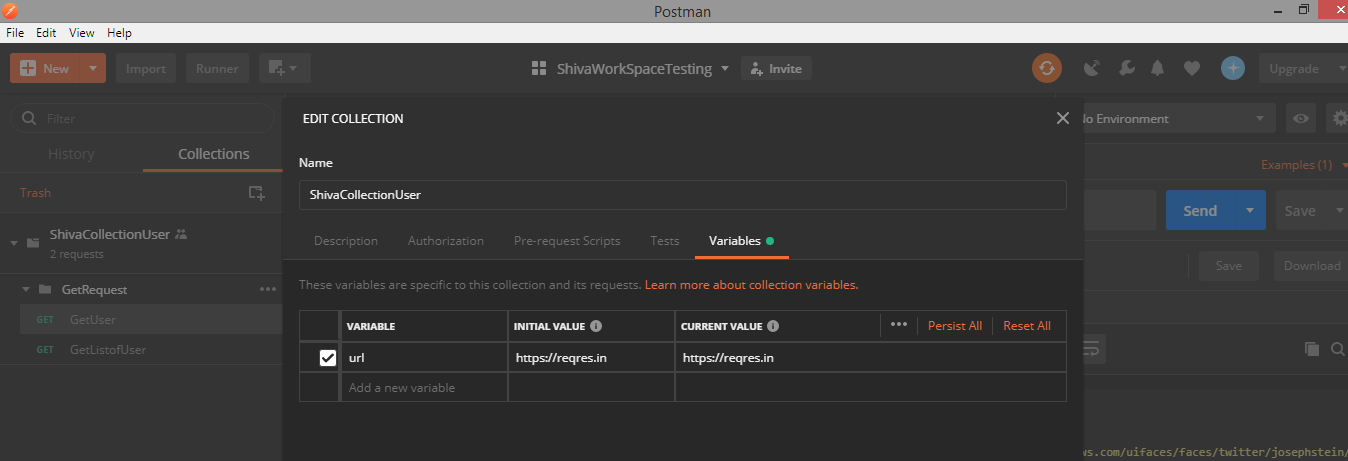


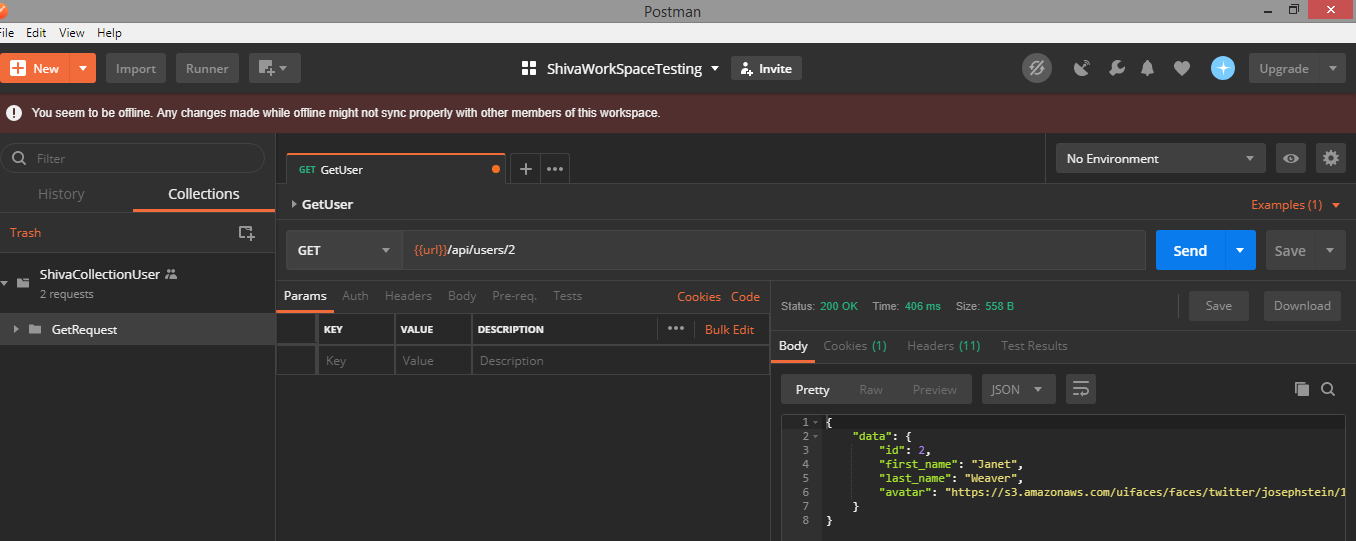
Collection Run:



Variables in API:

* Elements (Data store) that can take different values.
* Use of variables :
  + Reuse values at multiple places.
  + Avoid repetition
  + Avoid rework, when values are changed.
  + Variables can be set for collections and Environment level.
  + Variables are inserted in double flower brackets {{variableName}}





Set and get variables using scripts:

let urlvalue = pm.environment.get(url);

console.log(urlvalue);

* Diff between patch and put HTTP method

Suppose we have a resource that holds the first name and last name of a person.

If we want to change the first name then we send a **put** request for Update

{ "first": "Michael", "last": "Angelo" }

Here, although we are only changing the first name, with PUT request we have to send both parameters first and last.  
In other words, it is mandatory to send all values again, the full payload.

When we send a PATCH request, however, we only send the data which we want to update. In other words, we only send the first name to update, no need to send the last name.

REST CLIENT TESTING

POST :

