 **---------------------------------------------------------------------**

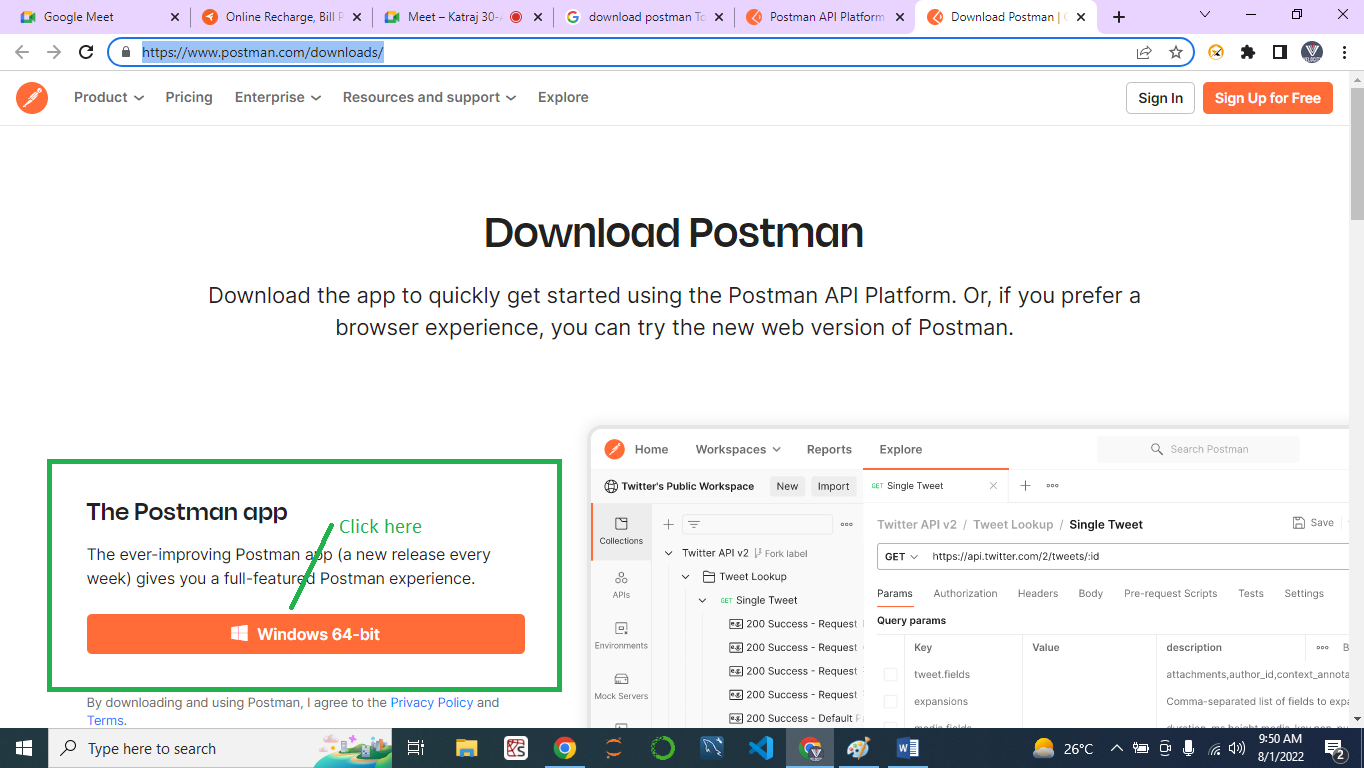
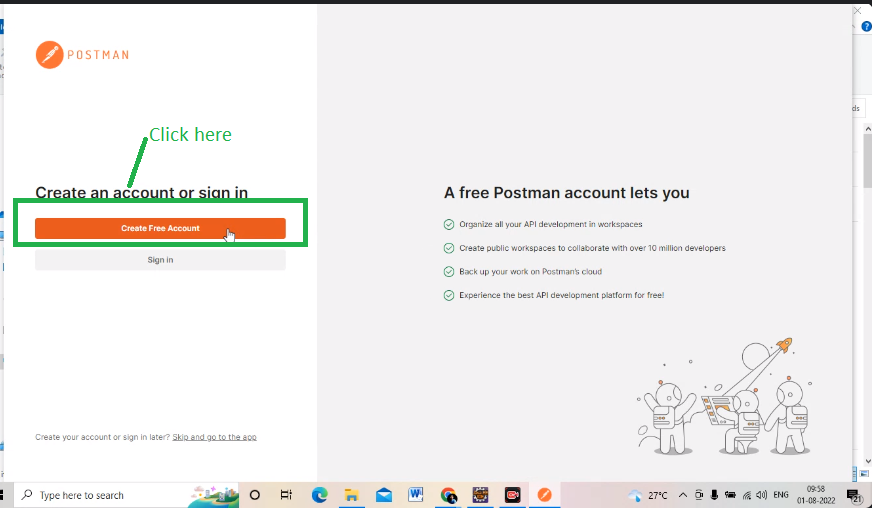
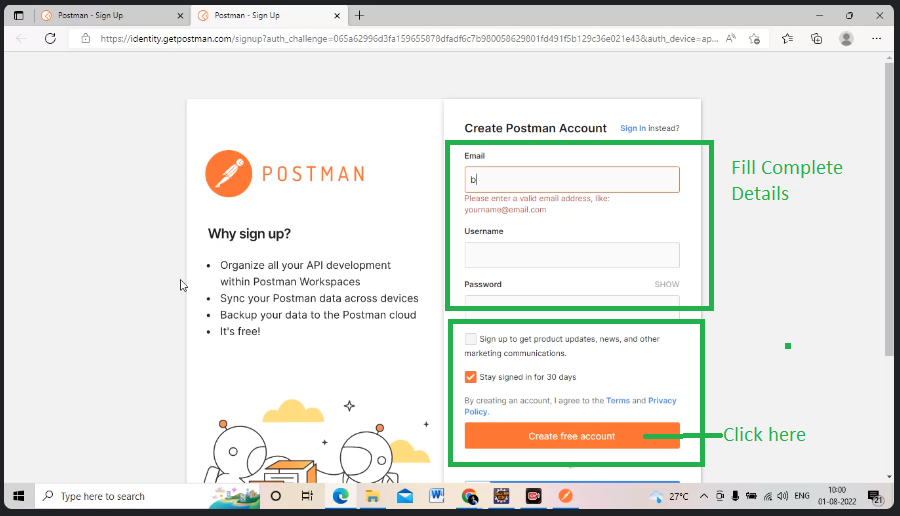
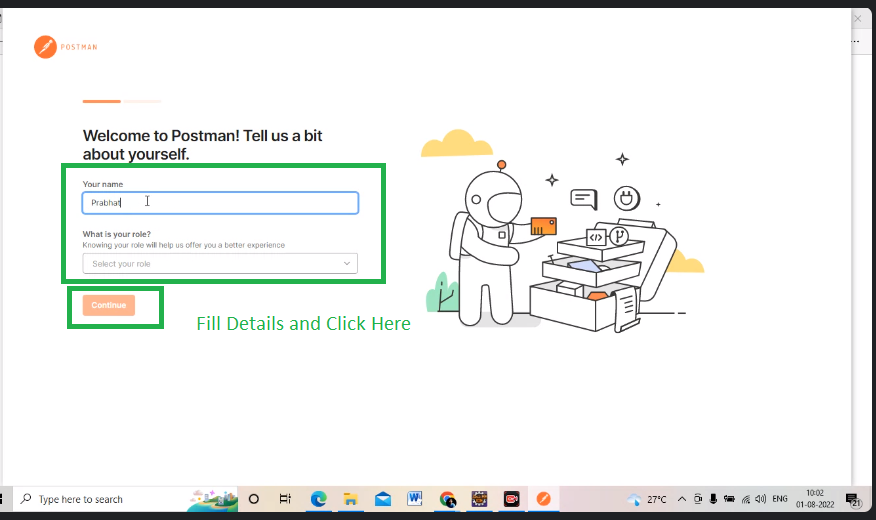
**Learn API by Vaibhav Sir**

**---------------------------------------------------------------------**

**Contents Page No.**

1. POSTMAN Installation 01
2. API Theory 04
3. Types of REST services Methods 08
4. Status Codes 09
5. Postman Tool 10
6. Test Case (GET method) 11

**POSTMAN Installation:**

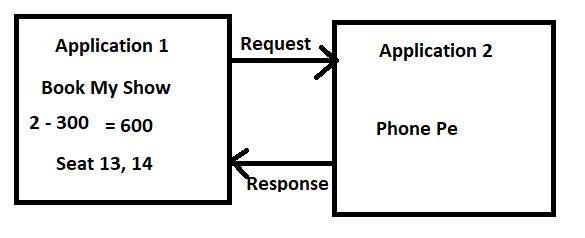
1. [**https://www.postman.com/downloads/**](https://www.postman.com/downloads/)
2. ****
3. **Double Click on the Postman Exe file.**
4. ****
5. ****
6. **Open the Postman App**
7. ****

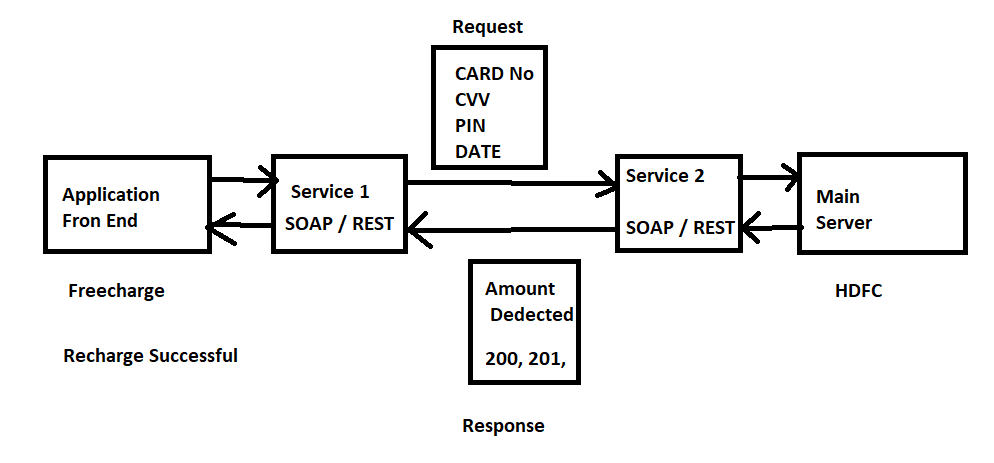
**API Testing / Web Service Testing. (Manually or Automation)**

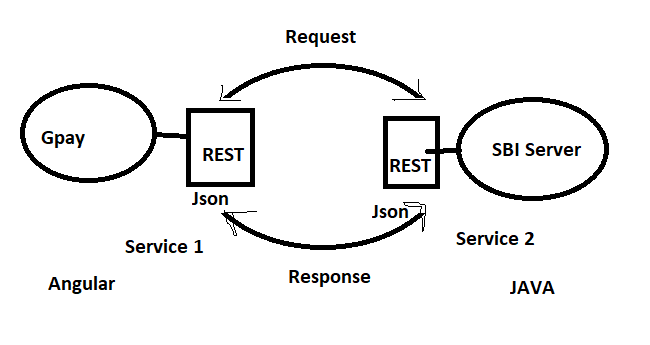
* API – Application Program Interface.
* API is software intermediary that allows applications to communicate with each other.

**Web service communication**

* It is communication between the End User and the application or Communication between the two applications.
* When User click on the Enter button (of Recharge) at that time a **Request is sent** by the Service 1 and when request is sent User / application get the **Response from the Service 2.**

****

****

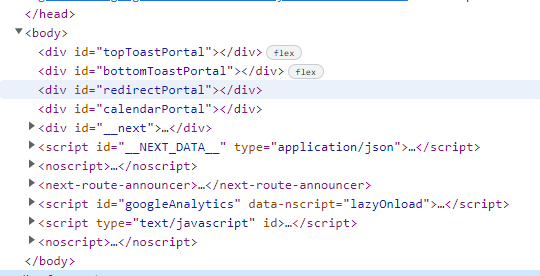
****

|  |  |
| --- | --- |
| **Manual Testing** | **Web Service Testing/ API Testing** |
| 1. **UI - Present** | 1. **UI – Not Required** |
| 1. **System and Functional /Smoke /Regression /Retesting** | 1. **Only Functional Testing (Request and Response)** |
| 1. **Application / Build Required** | 1. **URL / URI (Rest Services) , WSDL(SOAP)** |

**Important Terms.**

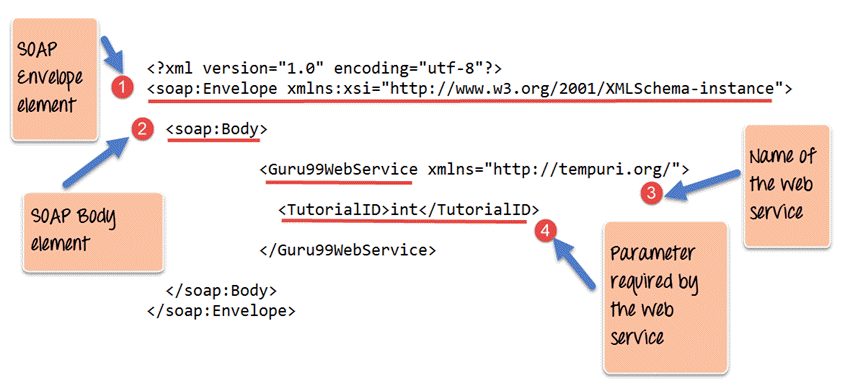
1. **XML – Extreme Mark Up language.**

**(HTML, Header, Body, div, fault)**

****

1. **SOAP – Simple Object Access Protocol**

**(Envelope, body, Parameters, Header, fault, div)**



1. **WSDL – Web Service Descriptive Language.**

**(Data type, Elements, message, binding)**

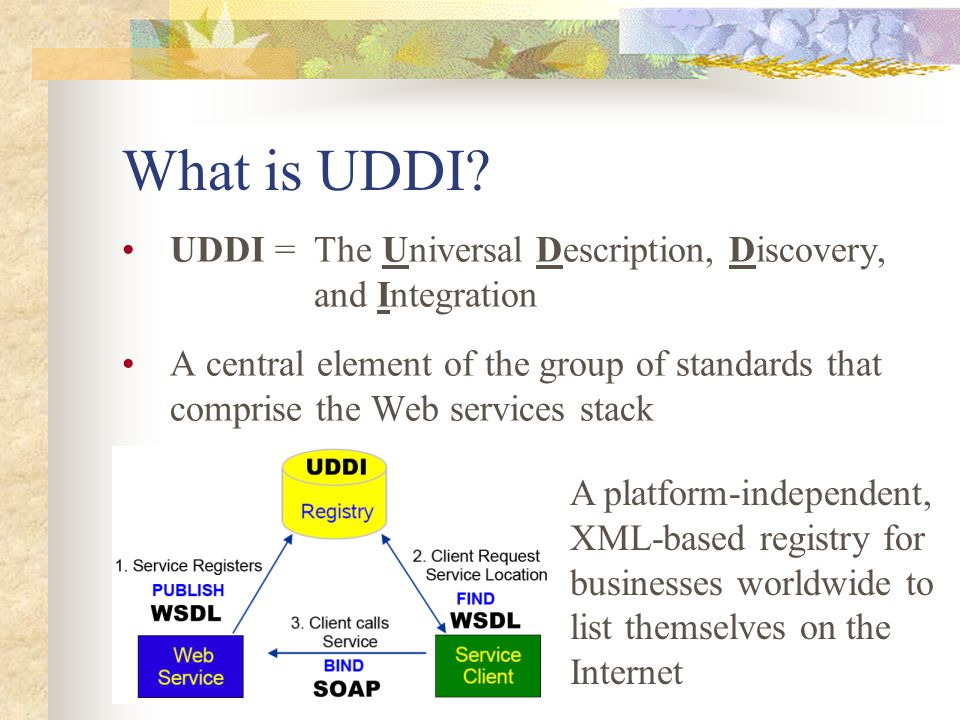
* WSDL is **an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information**

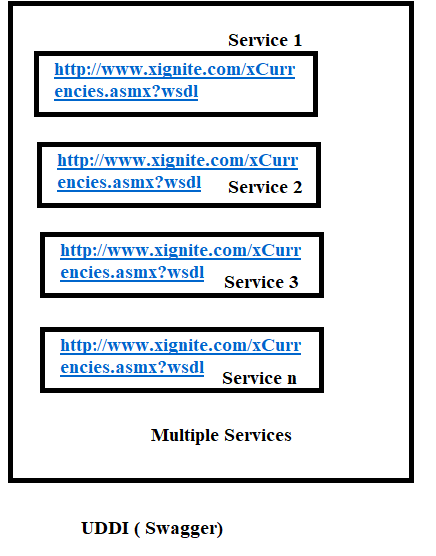
**Eg -** [**http://www.xignite.com/xCurrencies.asmx?wsdl**](http://www.xignite.com/xCurrencies.asmx?wsdl)

1. **UDDI – Universal Descriptive, Discovery and Integration**

**(WSDL file Repository)**

[**https://petstore.swagger.io/**](https://petstore.swagger.io/)



****

**----------------------------------------------------------------------------------------------------**

**SOAP**

**Envelope – SOAP Message (Either Request or Response)**

**Header – Authorization token, Access, i.e Username Password**

**Body – main parts of the SOAP message (E.g – When Search for the Travel then all the Travel display that names are present in body)**

**Fault – error message.**

**---------------------------------------------------------------------------------------------------**

**WSDL**

**Data Type – Type of Data which is used e.g int, float, string, etc**

**Elements – Operation perform on the data.**

**Message – Functionality of the Service**

**Binding – Combine all the functionality**

**--------------------------------------------------------------------------------------------------**

**There are two types of the Services**

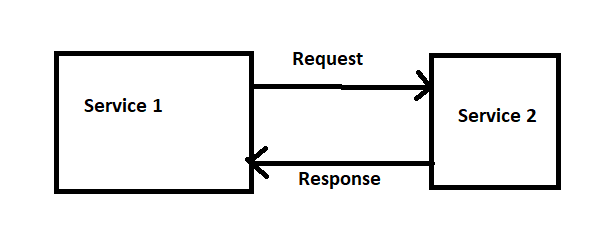
1. **SOAP Services – Simple Object Access Protocol.**
2. **REST Services - Representational State Transfer**.

|  |  |
| --- | --- |
| **SOAP Service** | **REST service** |
| 1. **SOAP is a Protocol.** | 1. **REST is an Architecture. (Special Design of Group of protocol)** |
| 1. **SOAP services used for only web based application.** | 1. **REST services used for Web based, mobile based and standalone application.** |
| 1. **Language – Required Only XML** | 1. **Langauge – HTTP/ HTTPS / URL /URI / Json** |
| 1. **For testing; WSDL file** | 1. **For testing; URL/URI** |
| 1. **SOAP services are slow because of the High Bandwidth.** | 1. **REST Services are Fast because of low bandwidth.** |

**---------------------------------------------------------------------------------------------------------------------**

**Common Test cases used for API Testing.**

1. **To validate the REST Response.**
2. **To validate the Data and Count in the response.**
3. **To validate the Tagname / Attribute present in the response.**
4. **To validate the status code in the response.**
5. **To validate the time taken for the response.**
6. **To validate the Assertions applied for the Verification.**
7. **To validate the functionality by passing test data.**
8. **To validate the functionality by scenarios.**

**REST Services**

[**https://petstore.swagger.io/**](https://petstore.swagger.io/)

**Collection –** Collection is bunch of request (Request stores in Collection

Type of Rest Services (Request) Methods

1. GET: When S1 Request S2 🡪 to get the data of the S2.

E.g – Phone Pe (S1) 🡪 Bank Account HDFC (S2): View Balance -🡪 Respose - RS. 1200 (UI of Phone Pe).

(SQL – Parallel Command: Select)

1. POST : When S1 Request S2 🡪 to Create a data in service 2. (to create)

E.g – Facebook Profile Pic Upload/Post.

(SQL – Parallel Command: Insert / Create)

1. PATCH : When S1 Request S2 🡪 to update the Data of S2 (Which already stored)  
   (SQL – Parallel Command: Update)
2. PUT : When S1 Request S2 🡪 to update the Data of S2 (Which already stored)

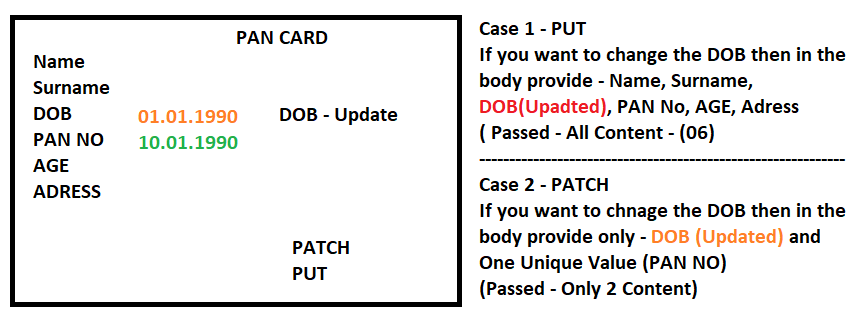
(SQL – Parallel Command: Update)

1. DELETE

When S1 Request S2 🡪 Delete the record / data from s2.

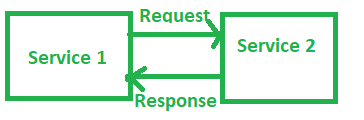
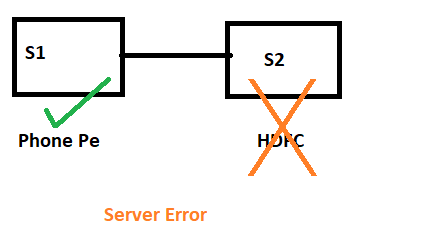
(SQL – Parallel Command: DELETE)

(PATCH method is Fast one)

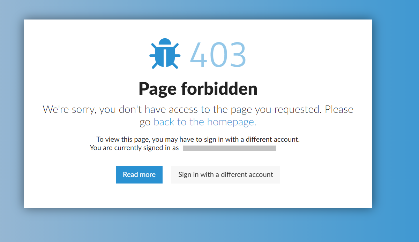
****

**Status Code**

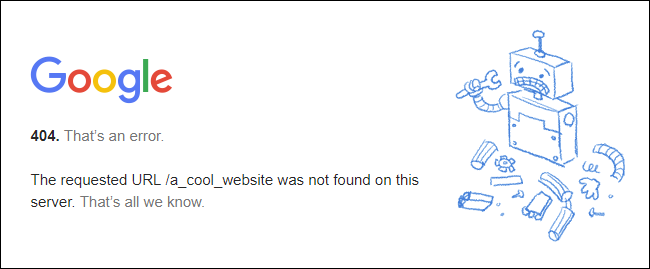
**Developer define the Status Code.**

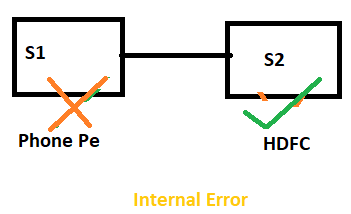
1. **Successful Response – 2XX (200, 201, 202, 204)**
2. **Server Error – 4XX (400, 401, 403, 404, 405)**
3. **Internal Error - 5XX (500, 501, 503)**
4. **Successful Response.**
5. **200: OK – GET: Request Successful and used get method.**
6. **201: Created – POST: Request Successful for Create when used POST method.**
7. **202: Accepted – PATCH/PUT: Request for the Update S2 and Request successful when used PATCH or PUT method.**
8. **204: No Content – Delete: When request sent to S2; to delete the Data🡪 Delete Successful using DELETE method.**
9. **Server Error**

**S1 send the request to s2 but there is on issue at s2.**

1. **400: Bad Request: When s1 request s2, if invalid data is show.**
2. **401: Unauthorized: When request sent by s1; S1 sent Wrong API Key / Invalidate Authorization request to s2.**
3. **403 : Forbidden : If you don’t have access /**

**When s1 request s2; one page depend on another page and 1st page closed.**

1. **404 : Not Found : s1 --- Request – S2 ; the content is not available on the s2; Wrong URL, Server can’t find the request resource**
2. **405: Method not allowed – Instead of get method you select the PUT method.**

****

1. **Internal Error**

When s1 us not working but service 2 is working fine.

1. **500 – Internal Server Error**

**Any defect issue which service 1 unable to handle.**

1. **501 – Not Implemented – S1 request sent to s2; but the request yet not implemented.**
2. **503 – Service Unavailable – Service is not read at this moment or service will not work at this time.**

**( E.g – IRCTC Railway Tatkal ticket booking --- Service will open at 10 AM but you try at 9.00 AM)**

**POSTMAN TOOL**

**REST SERVICES –**

Swagger: <https://petstore.swagger.io/>

**GET Method / Request**

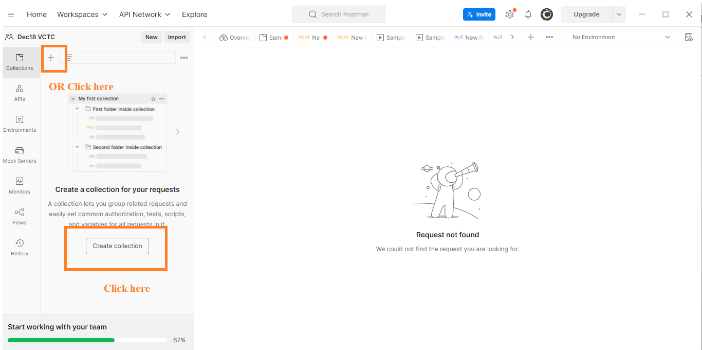
* Developer will provide URL /URI
* Rest Services URL/URI File - <https://reqres.in/api/users/2> or [**https://reqres.in/api/users?page=2**](https://reqres.in/api/users?page=2)
* Authentication Key / Barrier Token / Username Password –

Key - 1d05cba7905e926d170a0d656d168621

Value - 1d05cba7905e926d170a0d656d168621

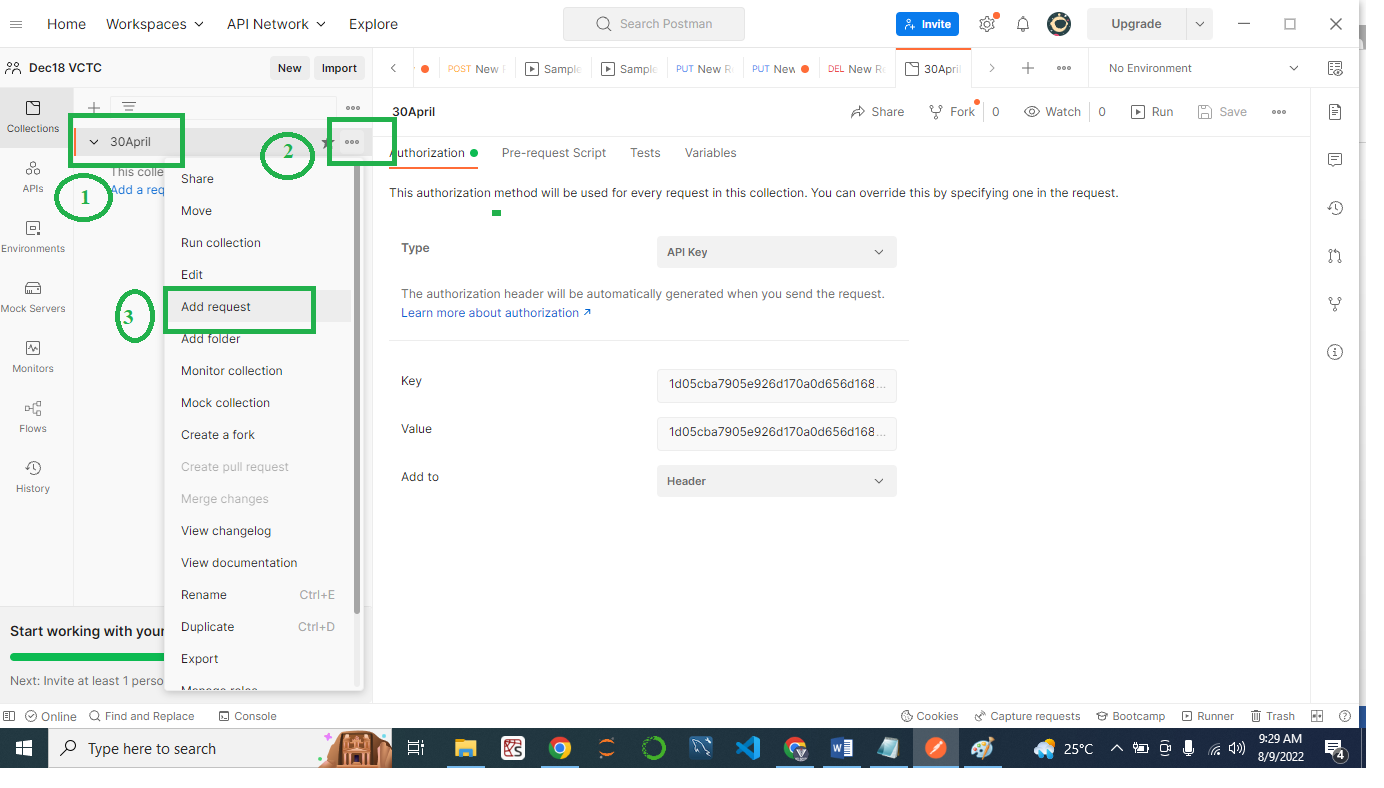
* **Time of response < 30 sec**

Step 1 : Click on Collection

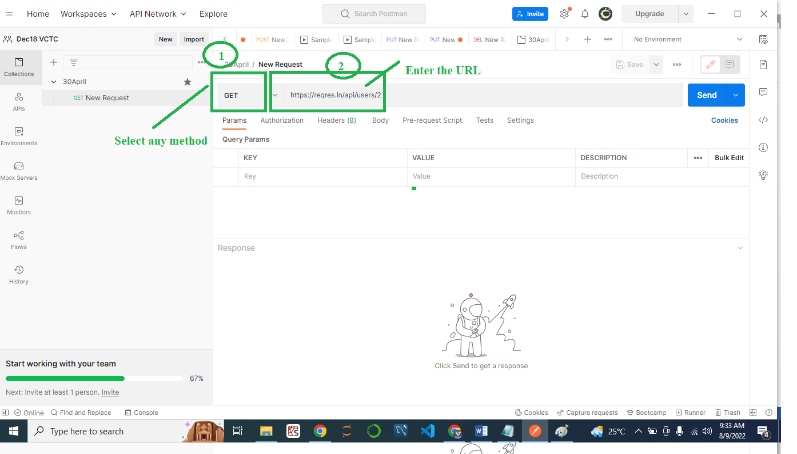


Step 2: Rename the Collection / Add API Key or Bearer Token or any authenticator.

Step 3 : Move (Hover Over) mouse on the Collection 🡪 Three dots visible 🡪 Click on three dots 🡪 Click on Add Request.



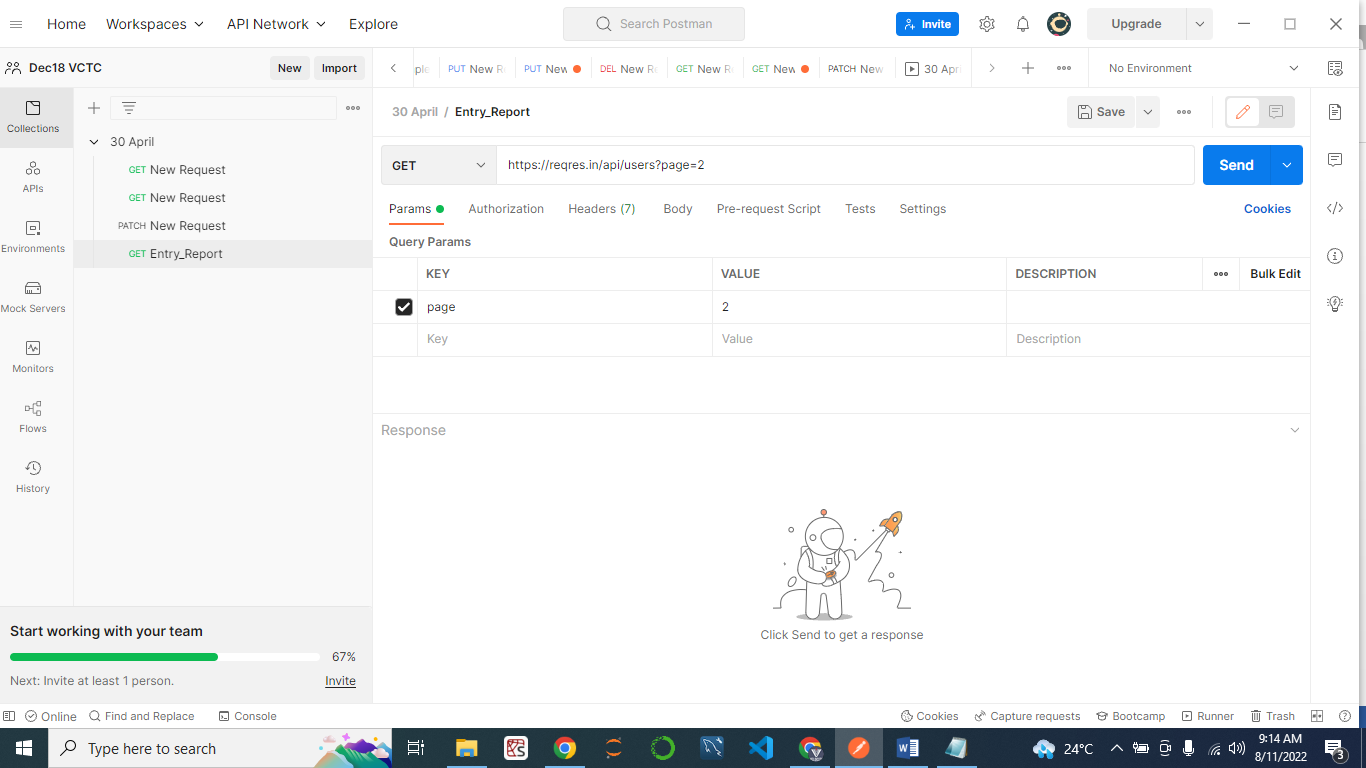
Step 4 : Select any request and provided URL

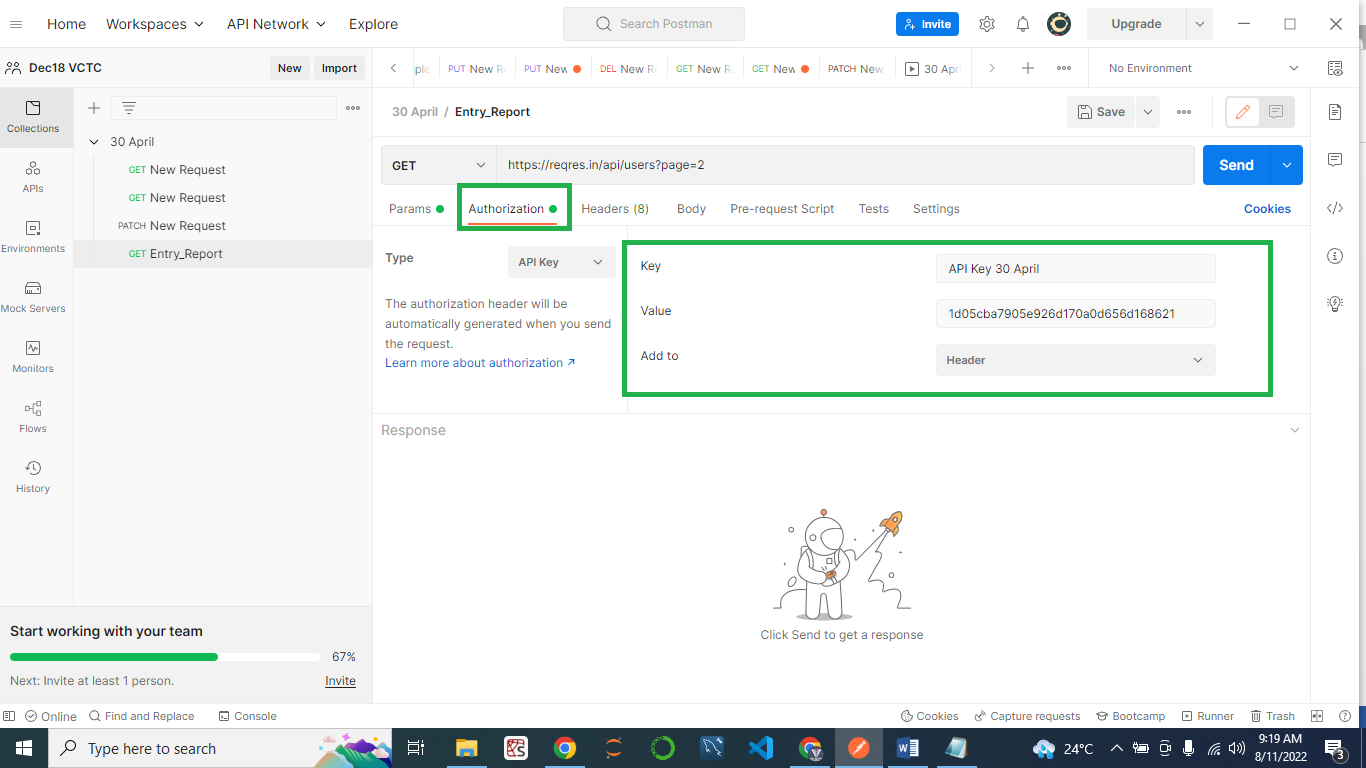


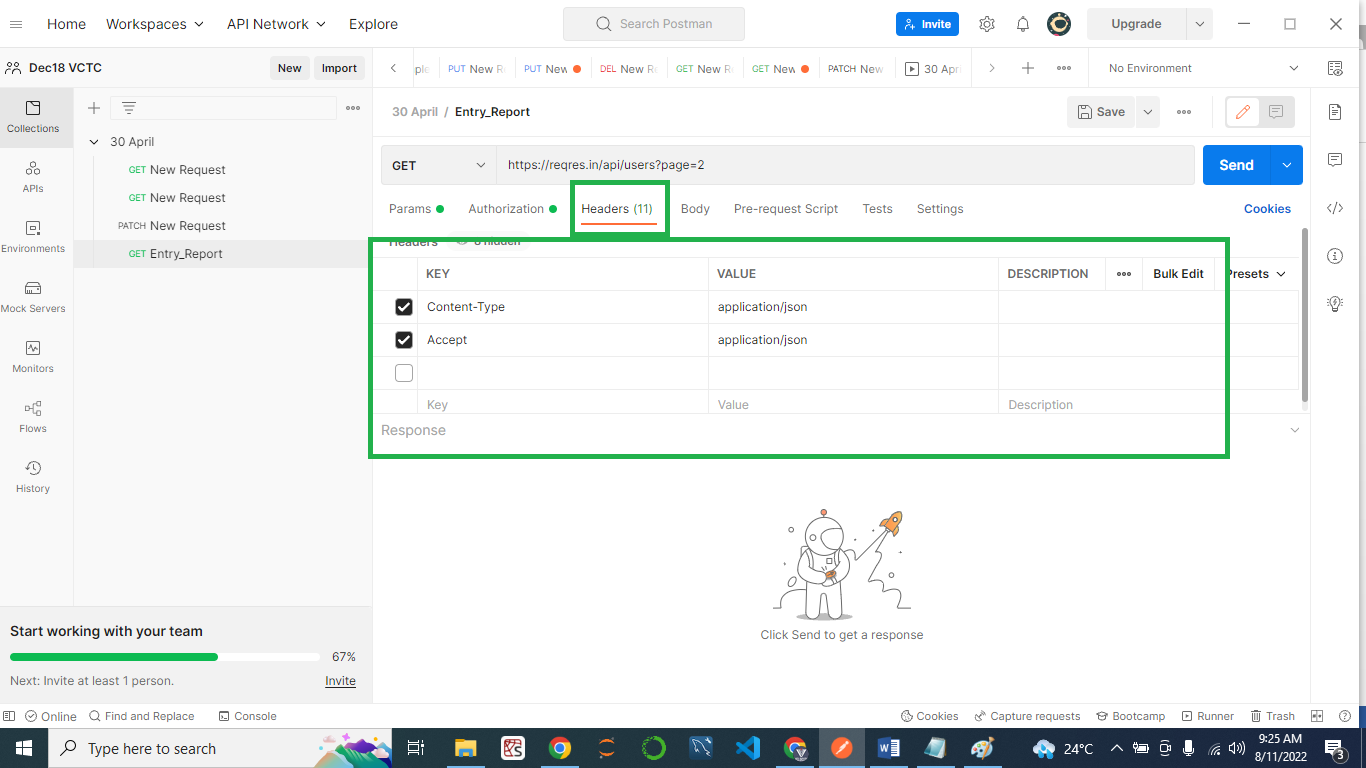
GET Method.  
<https://reqres.in/api/users?page=2>

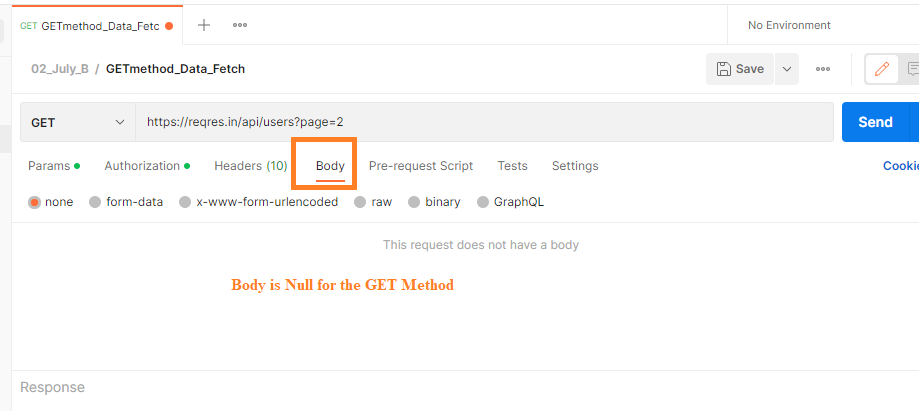
API Key - 1d05cba7905e926d170a0d656d168621

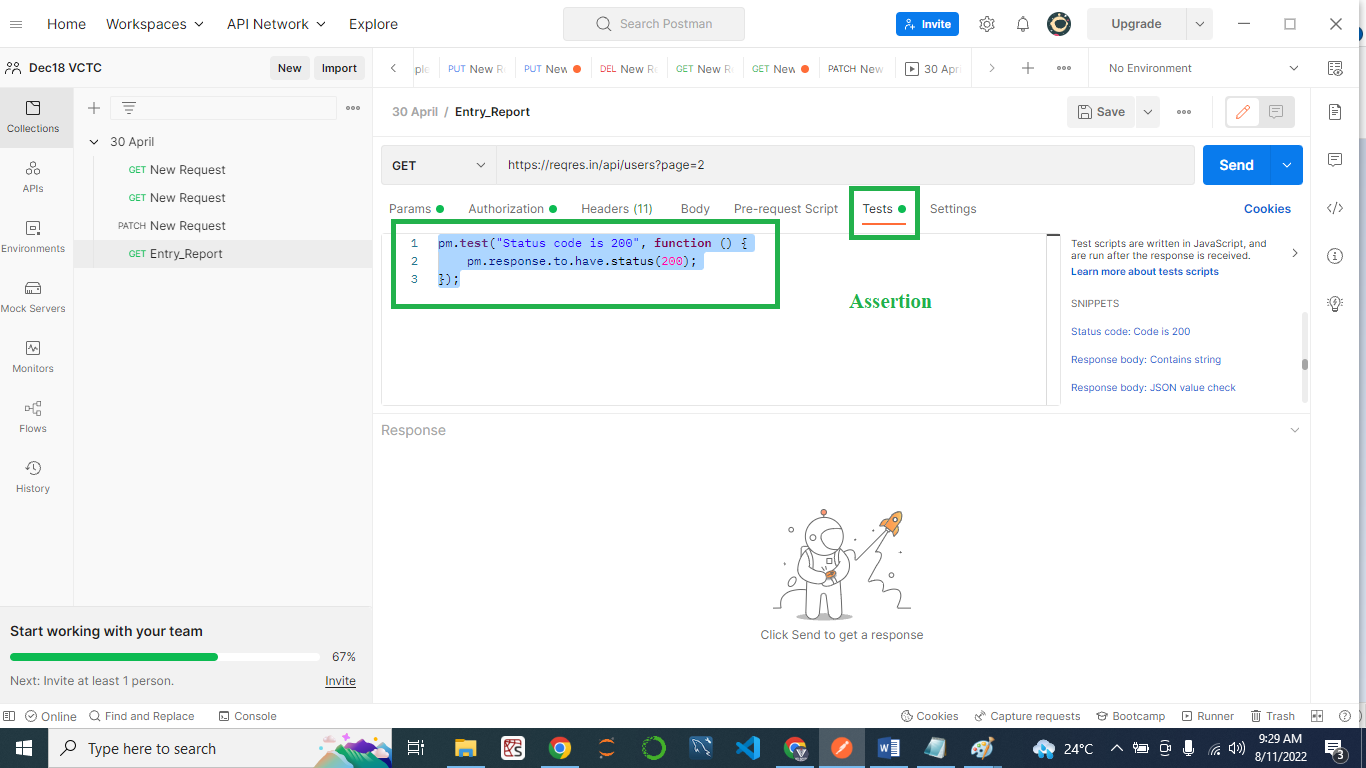
Time < 30 Sec





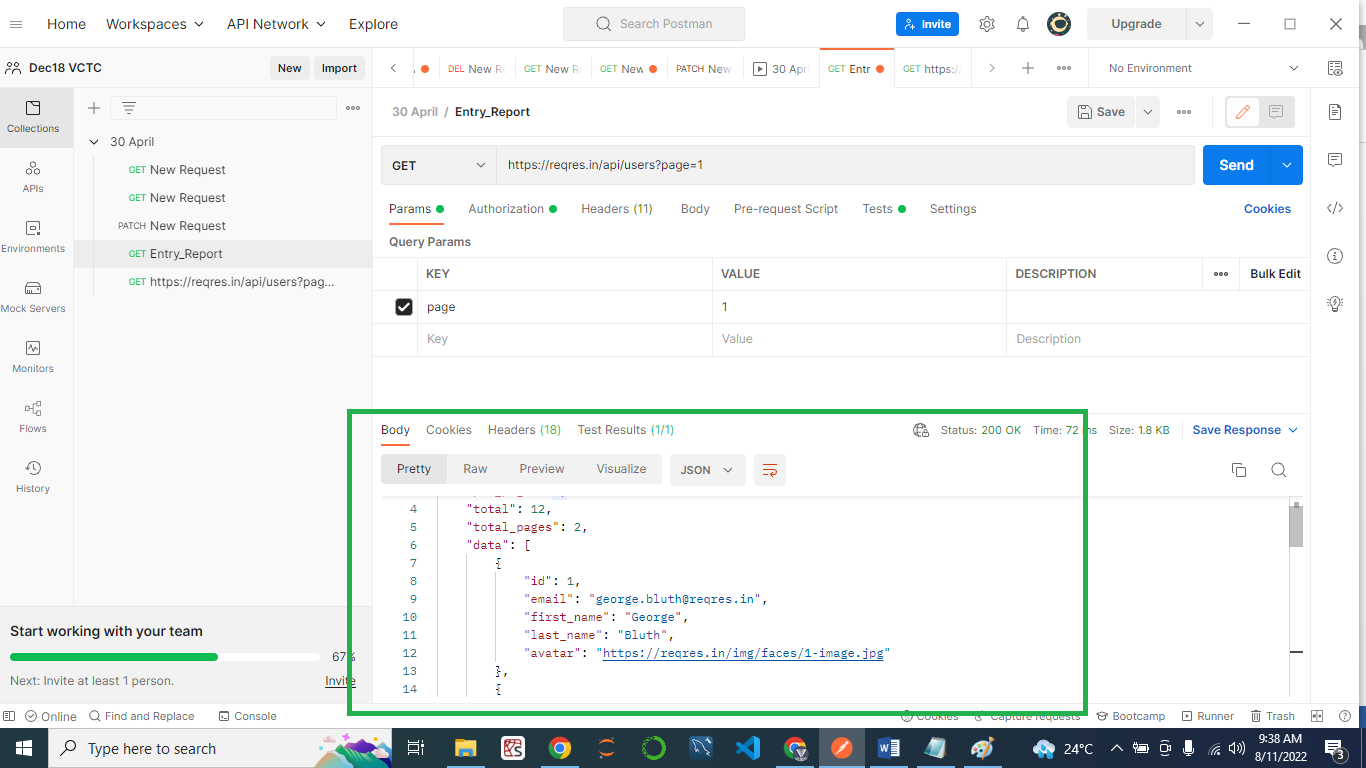




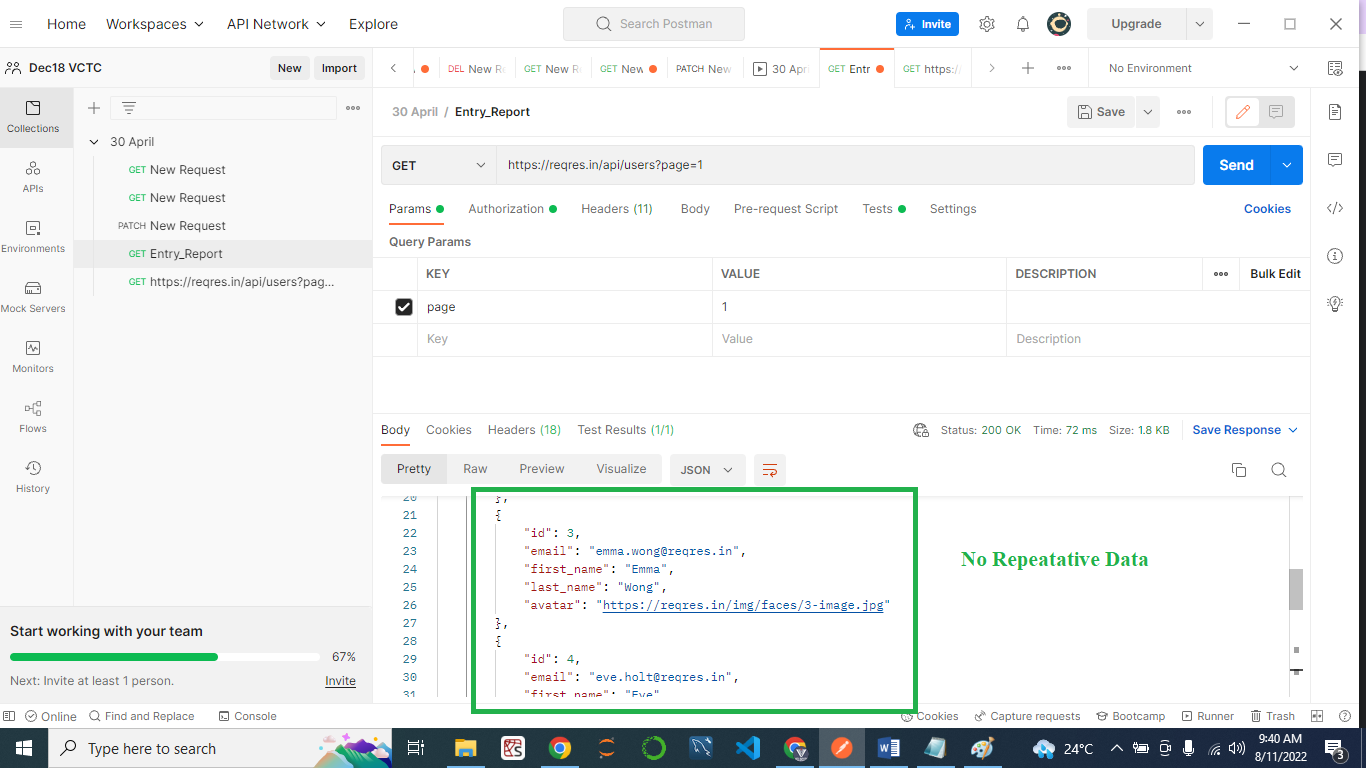


Testing API

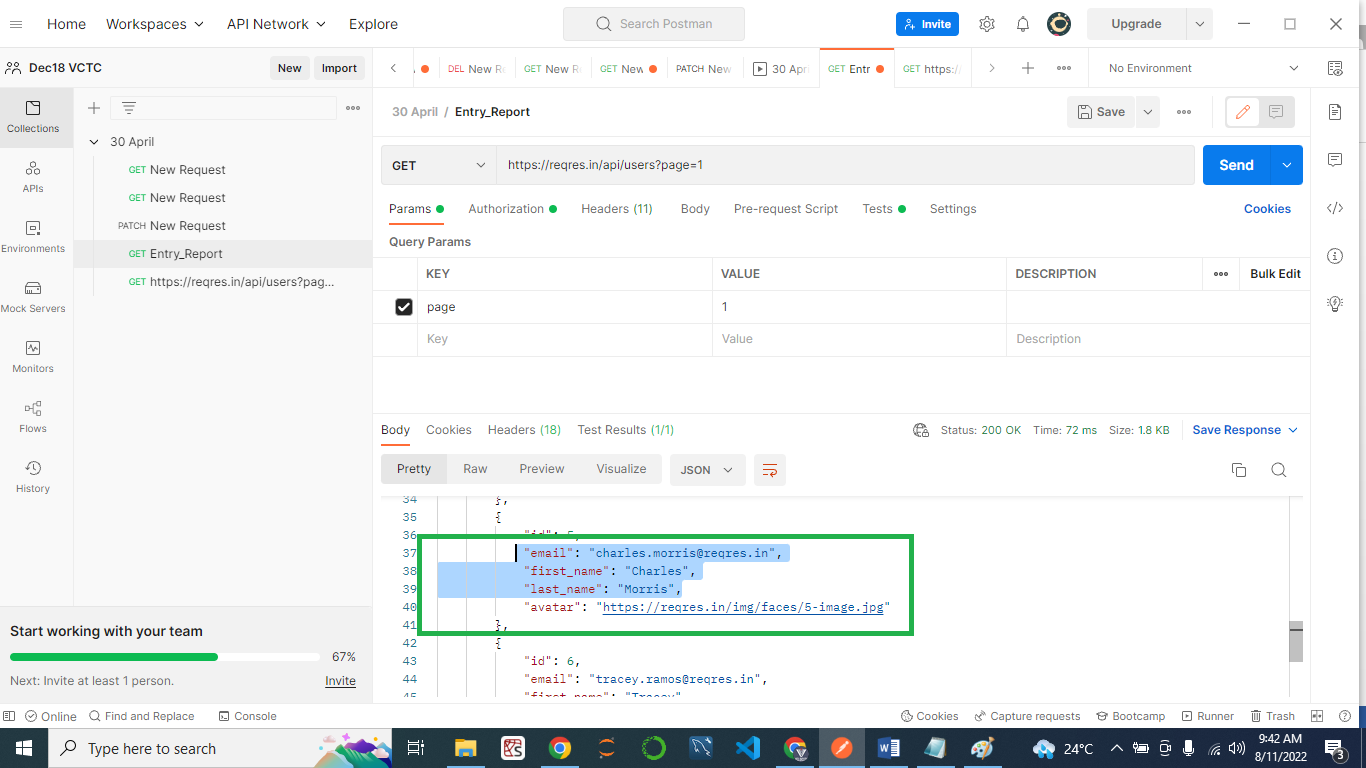
1. To verify Rest Service Response – PASS ( Get – 200 OK )



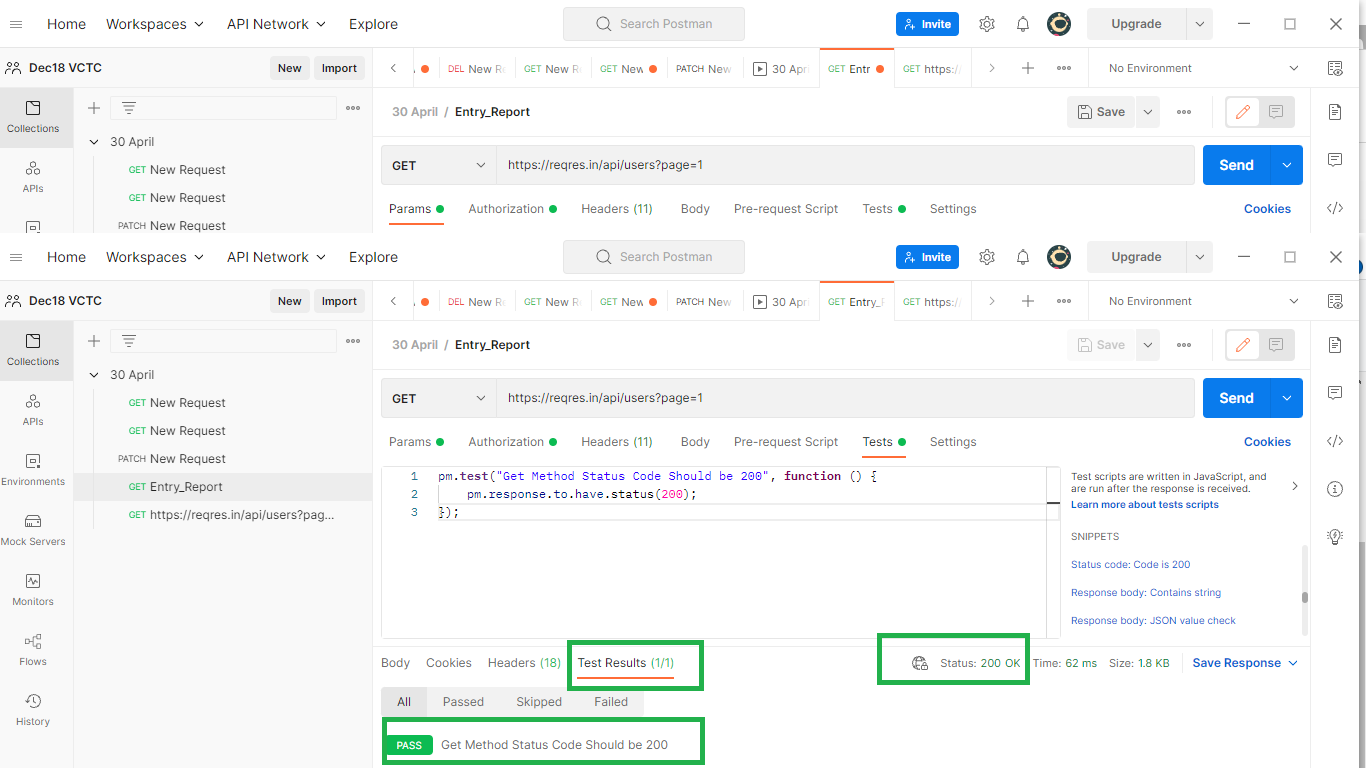
1. To verify Data and Count in response. – PASS (Data displayed only one time)



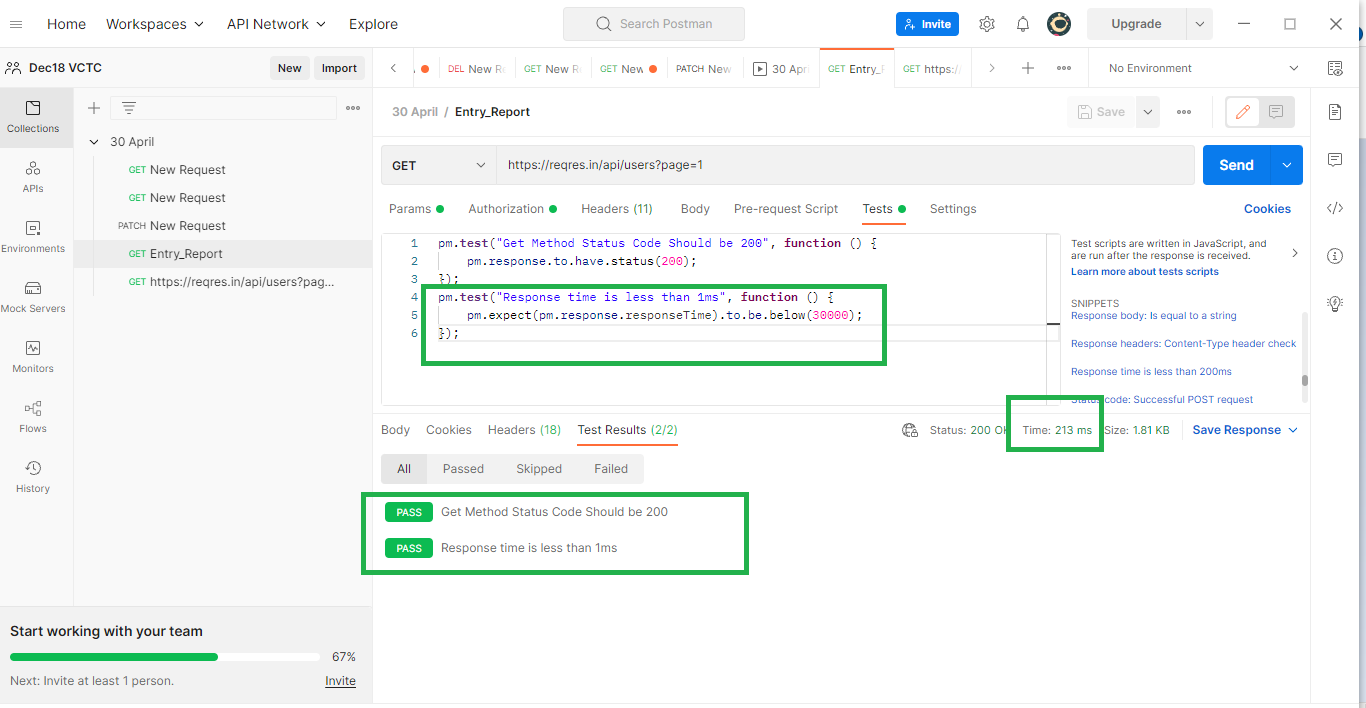
1. To verify Tagname and Attribute 🡪 PASS ( ID, Email, First Name, Last Name, Avatar)



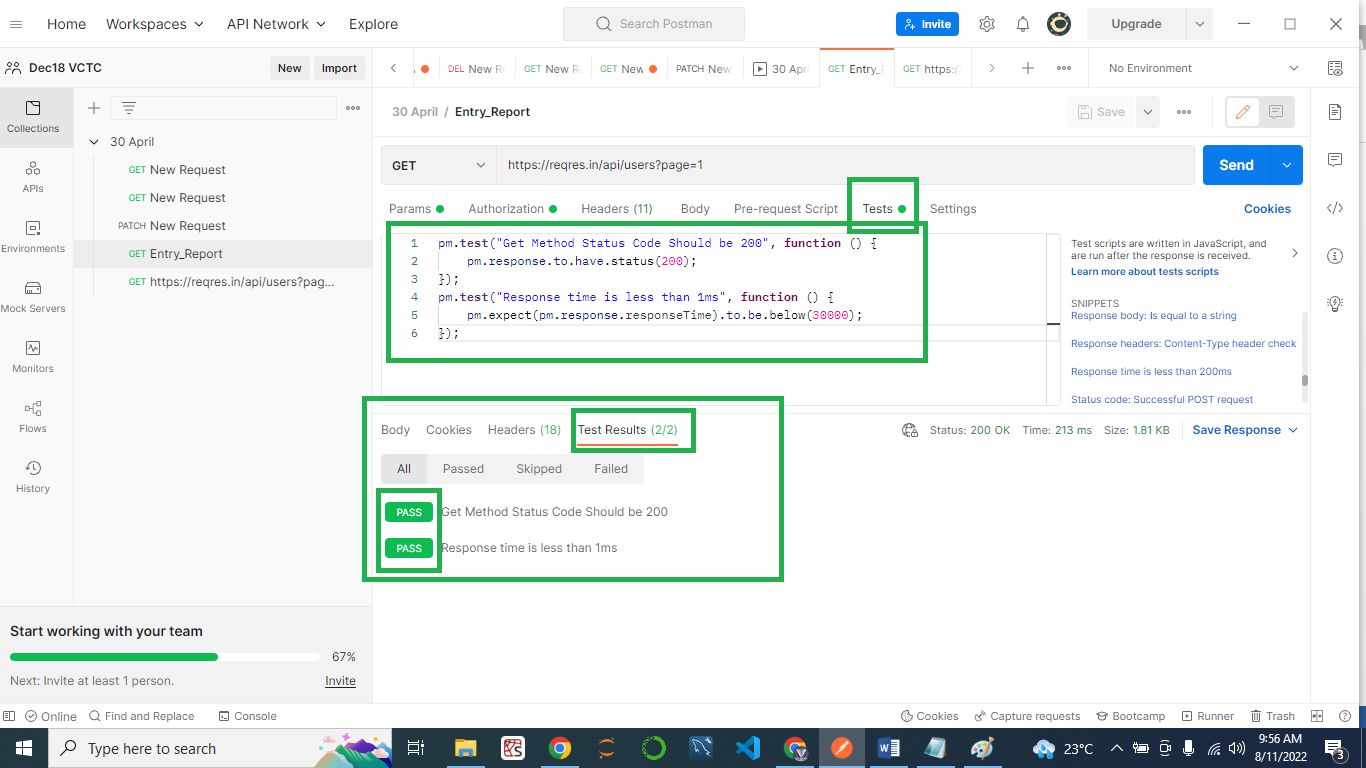
1. To verify Status code in the response. = PASS (Status Code : 200 OK)



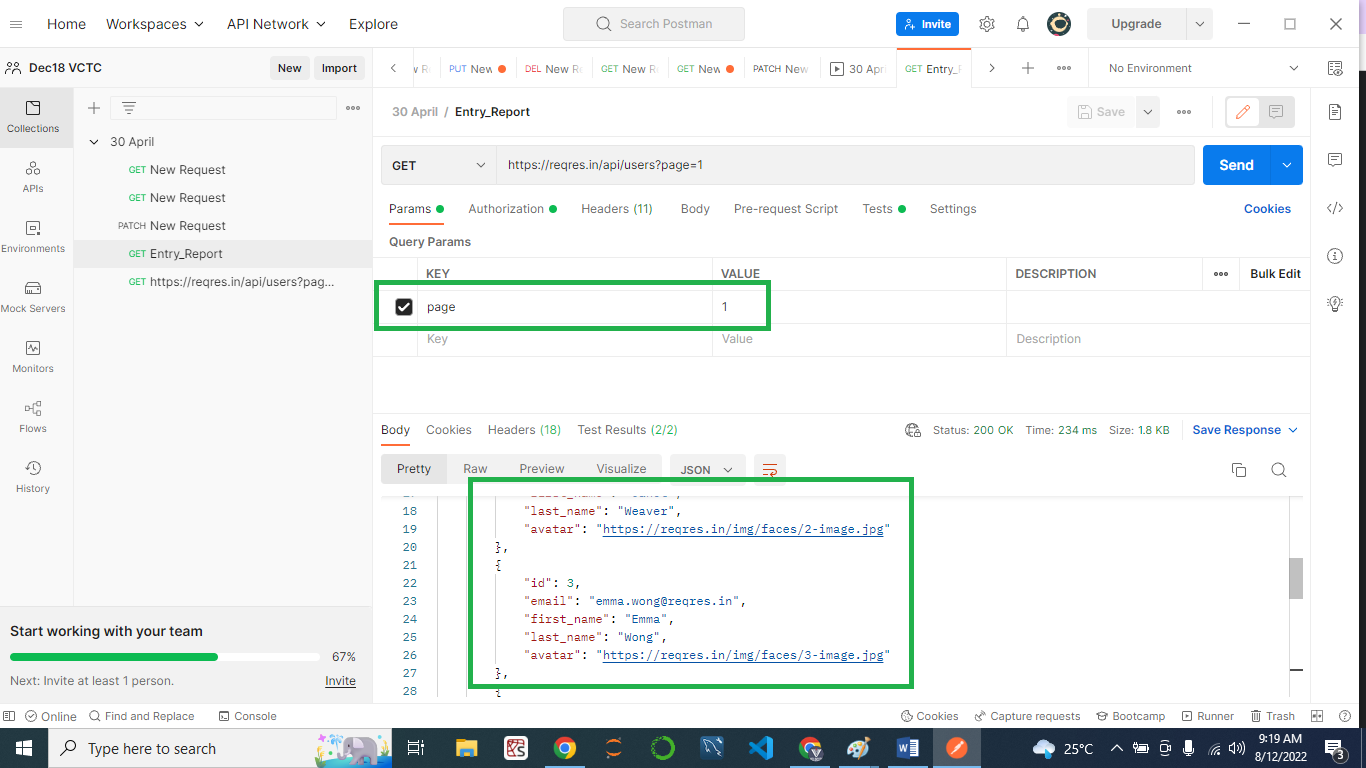
1. To verify time taken for the response 🡪 PASS ( Time of Response – 213 msec)

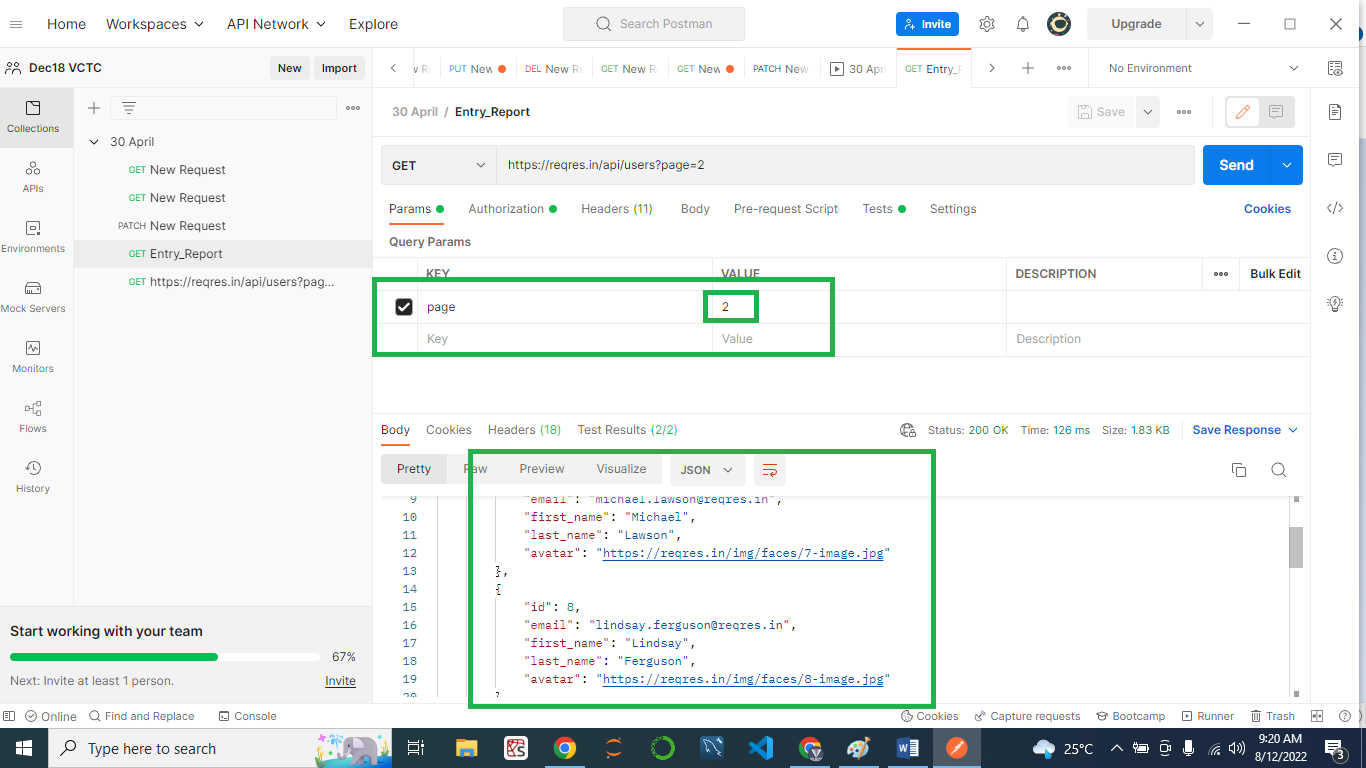


1. To verify the applied Assertion 🡪 PASS

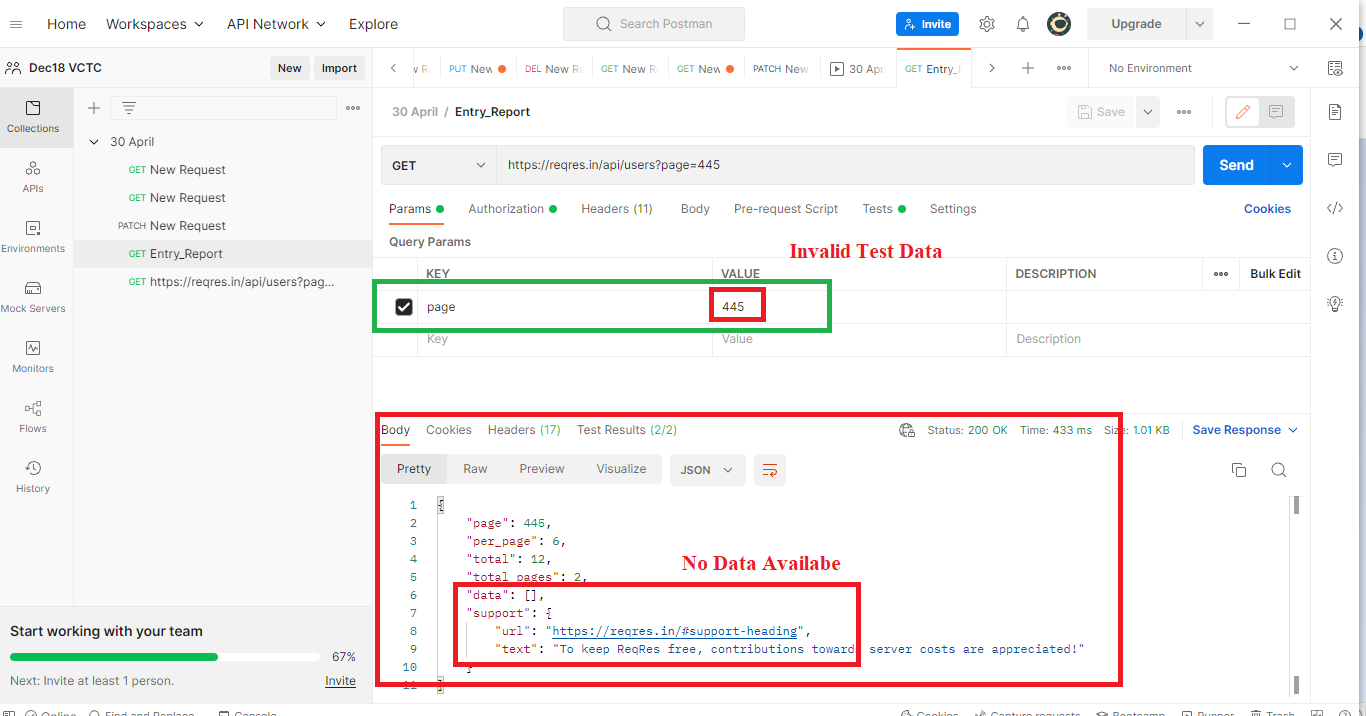


1. To verify the functionality by passing test data.





1. To verify the functionality by Negative Test Data



**POST METHOD**

* Dev will provide – URL URI <https://reqres.in/api/users>
* **Body / Payload**

**BODY**

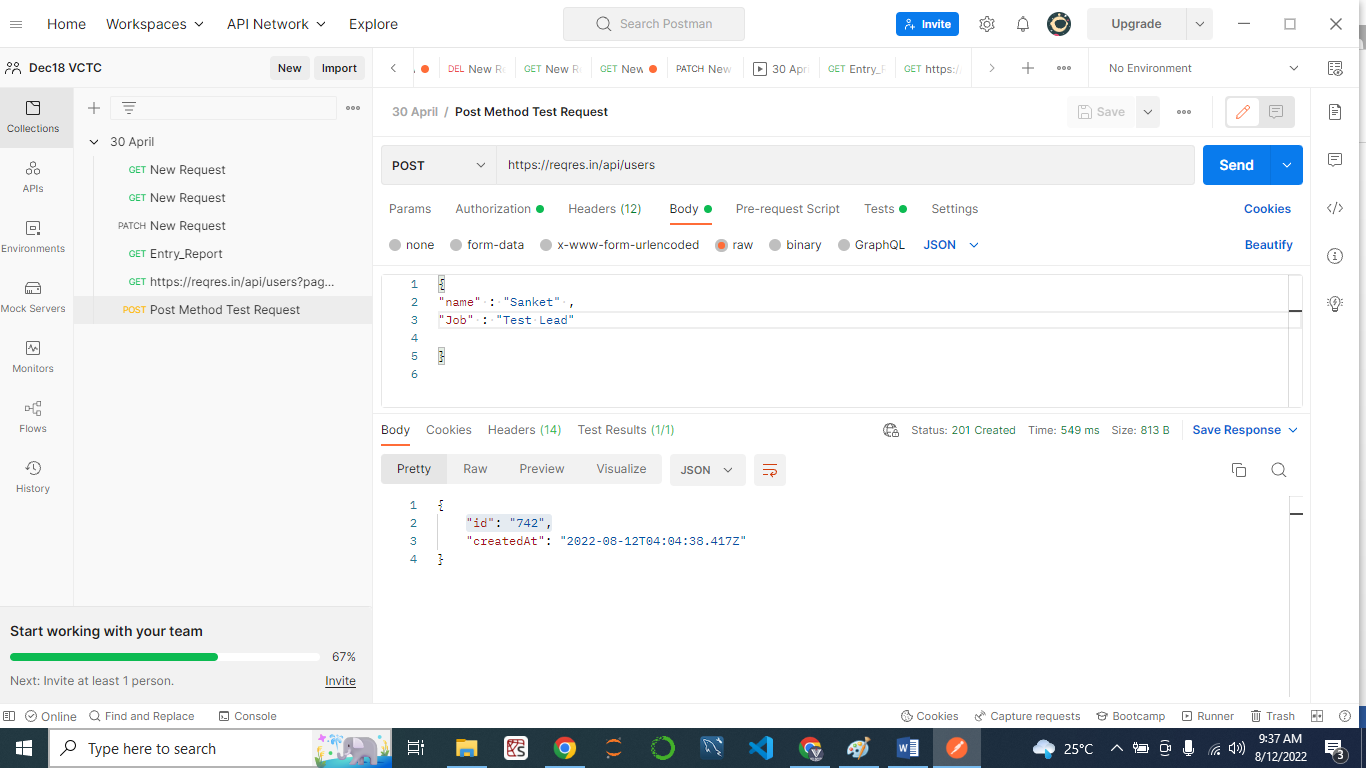
{

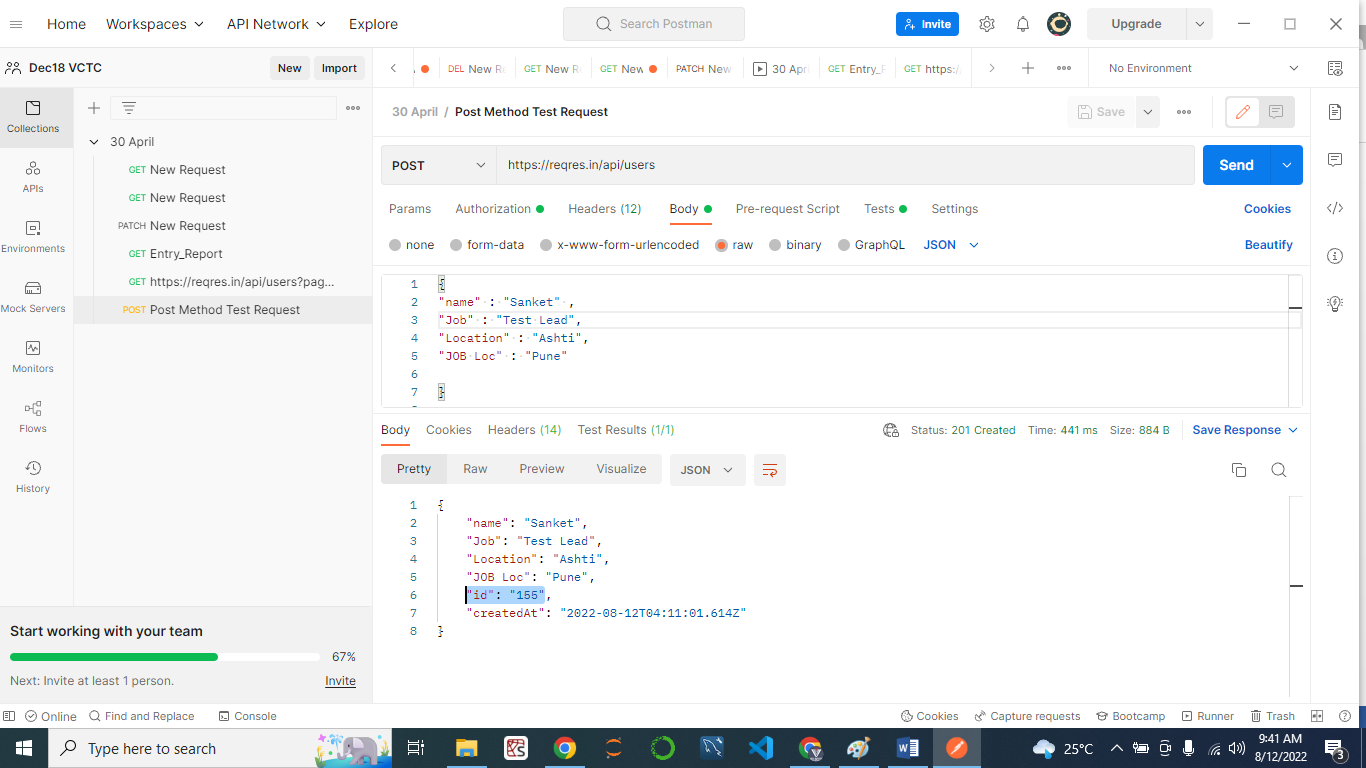
“name” : “Sanket”,

“Job” : “Leader”

}

* Authentication Key / Bearer Token / API Key
* Time Response < 30 Sec





Put Method:

**Dev Provide**

* URL - <https://reqres.in/api/users/2>
* BODY / PAYLOAD

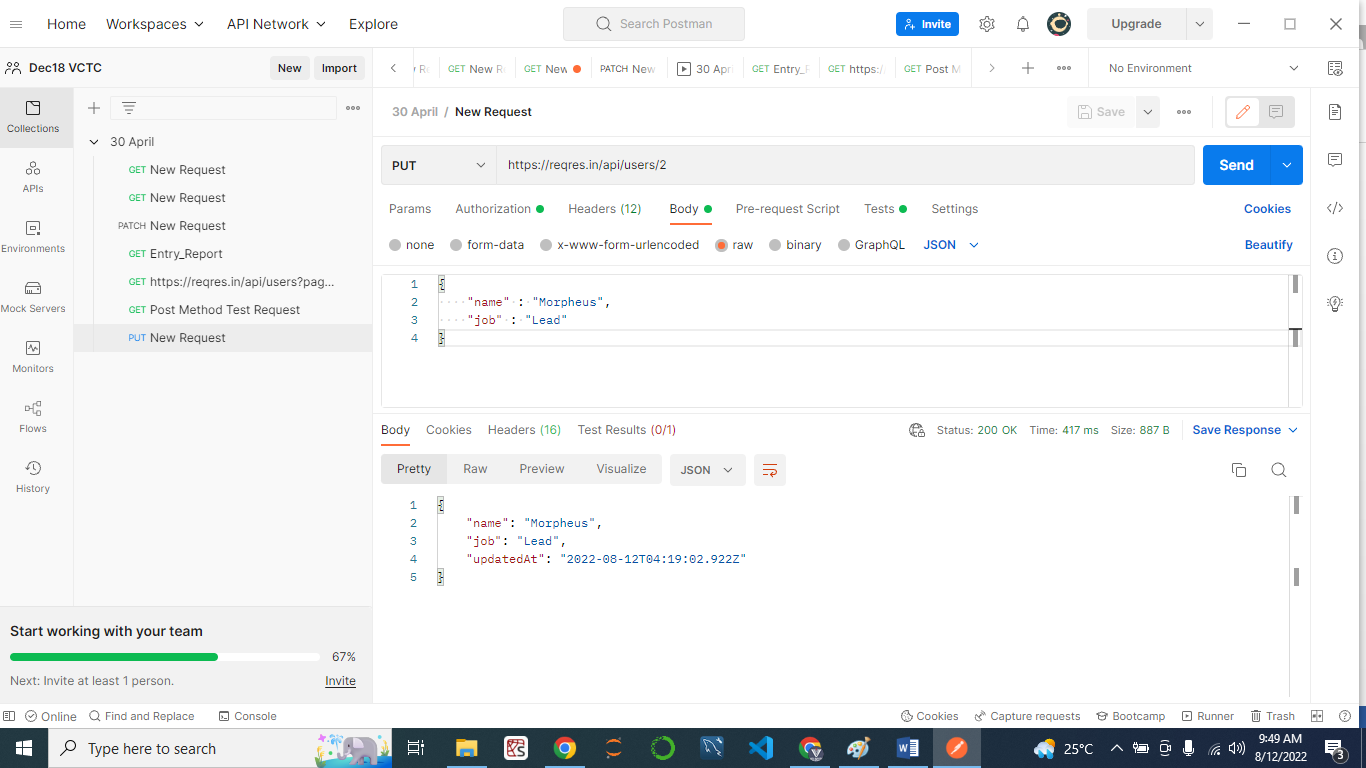
**{**

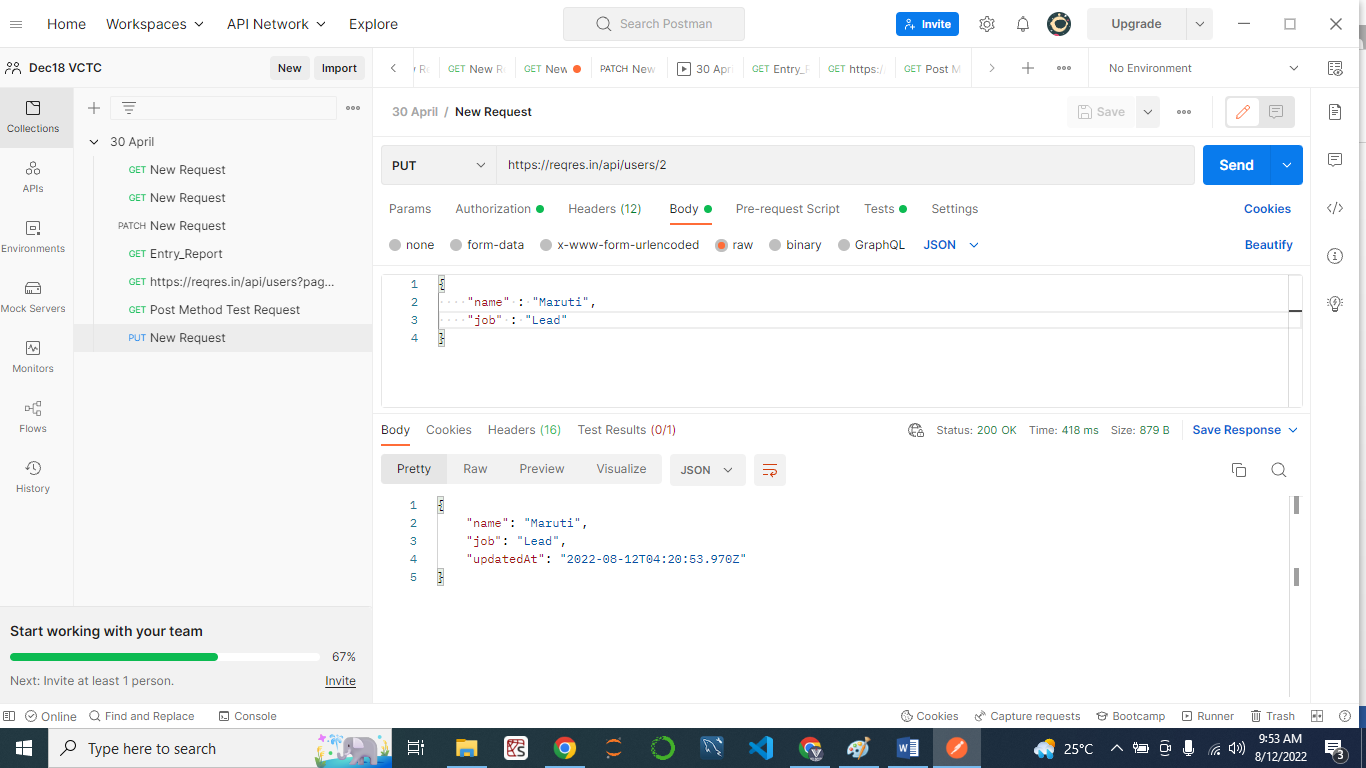
“name” : “Sanket”,

“job” : “Leader”

**}**

* **API Key / Bearer Token / Authentication Key**
* **Time of Response < 30 Sec.**





**API Templates**

[**https://postmarkapp.com/developer/api/templates-api#get-template**](https://postmarkapp.com/developer/api/templates-api#get-template)

### DELETE Method/Request-

* **Developer will provides -**
* REST Service- **URl/URI file**= <https://reqres.in/api/users/2>
* Authentication/ Barrier token/ Username & password = bdewafdfscvgbhgfsdf
* Time for response < 60 sec
* Unit Testing document- Steps for testing, URL/URI, Tables name, etc

## Real REST API Testing-

API Key-

GET request- URL- api.openweathermap.org/data/2.5/weather?q={city name}&appid=[{API key}](https://home.openweathermap.org/api_keys)

7. API Key Default : 1d05cba7905e926d170a0d656d168621

API Key weather : 0f54963d580a0ed846acdbbd63d3dfa0

https://openweathermap.org/current (URL of Page)

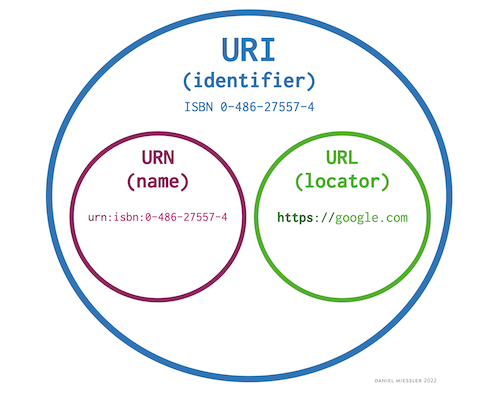
URI - api.openweathermap.org/data/2.5/weather?q={city name}&appid={API key}

**What is difference between Web service and API.**

**: All the web services are subset of the of API.**

**All webservices are API but All API are not webservices.**

|  |  |
| --- | --- |
| **Web Services** | **API** |
| All Web services are API | All API is not Web Services. |
| Web Services is subset of API | API is a Superset of Web services |
| SOAP / REST (Restriction – Mobile Based) | All format of Services |
| Need Network (WSDL) | Doesn’t need network. (URL/URI) |
| XML | JSON |

****

|  |  |
| --- | --- |
| **URL** | **URI** |
| Uniform Resource Locator | Universal Resource Identifier |
| URL is a subset of URI | URI is a superset of URL |
| URL locates on Webpage | URI Used in XML, HTML, Name or Location etc |
| Protocol – HTTP and HTTPS. | All forms of Protocol. |

**Q. What the different status codes used in your project.**

**Q. What are the different methods used for the API Testing in you project.**

**Q. Are you able to perform API Automation?**

**Well In my organization I haven’t got opportunity to work on API Automation but I have performed the API Testing Manually. I future if I got any opportunity to work on API Automation it would be a plus point to my career and growth.**

**Resume : https://www.canva.com/design/DAFQ2-gYTo8/gBBlATZ8vYLJmC04-wm1QQ/edit**