

## Learn API testing with Mr. Kapil Sir

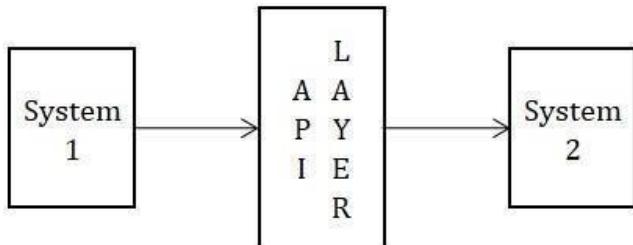
### API- Application Programming Interface.

- API is used to communicate between two systems.
- It is simply known as sending the request from one system to another system and getting the required response.
- For Ex. Communication between Amazon and gpay. Here we send request from amazon App to get the response as gpay.

### Advantages of API-

- **API provides the security.**

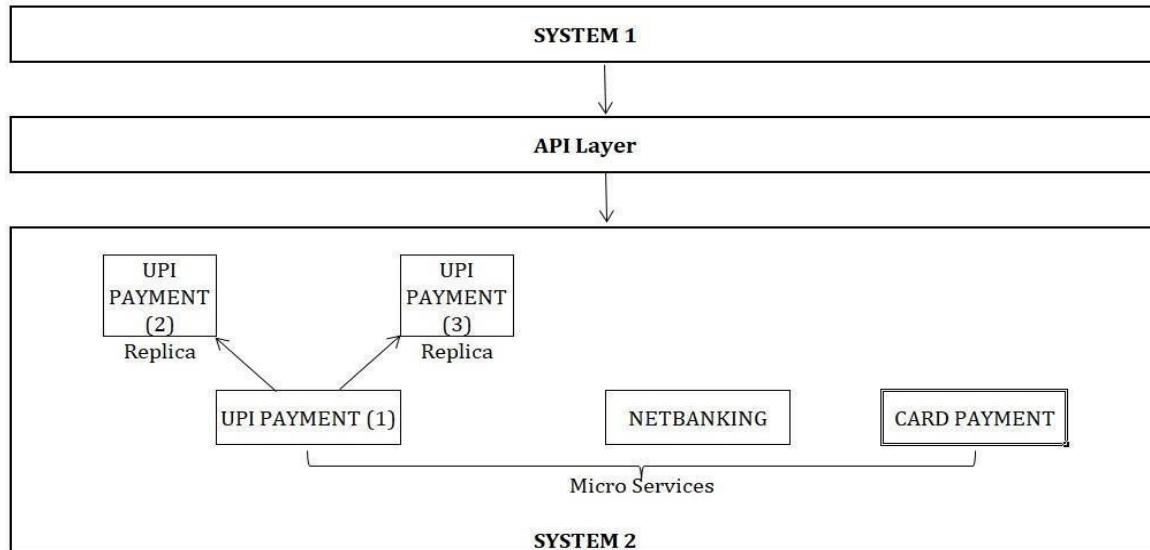
While communicating between two systems, API will provide API layer in between them which helps to secure our data.



- **To avoid the data breaching.**

As we know API provides the API layer for security purpose, So it also avoids Data hacking or breaching.

- **To increase the performance and Balances the load.**



In API, Developer already created a Replica's of Micro services which helps to increase the performance of the system by sharing the request with replicas in case of load on system 2 increases. Here API will decide how much load to be shared with each micro service and replica. As this balances the load, the performance will get automatically increases.

- **API helps in Data hiding.**

API helps to hide the data also.

- **API helps for Proper communication between two systems.**

When first system is sending the request to second system, then the request should go for second system only not the third one. For ex. User wants to do a payment for Amazon order by using Gpay app. Then the payment request should go for Gpay only. This is nothing but the proper communication. In between this communication process, API will provide API layer for Security purpose.

- **API also checks and authenticates the data which we are passing.**

- **API tests core functionality.**

- **API is time effective.**

We can hit lot of API's at a time with less time.

- **Language Independent.**

API is Language independent i. e API reads multiple languages like XML, JSON, HTML, TEXT etc.

- **Easy interaction with GUI.**

### **TYPES OF API-**

**1) REST API/SERVICES**- Uses POSTMAN tool(Representational state transfer)

**2) SOAP API/SERVICES** - Uses SOAPUI tool(simple object access protocol)

| <b>SOAP</b>  | <b>REST</b>  |
|--|--|
| SOAP is Protocol.                                      | REST is Architecture.  |
| Uses XML only.   | Uses XML, JSON, HTML, TEXT, JavaScript etc                           |
| SOAP reads WSDL file.                                  | REST reads API only.   |
| Type of security provided by SOAP is SOAP Environment. | Type of security provided by REST is AUTH Token, HEADER, PARAMS etc. |
| Heavy in weight API.                                   | Light in weight API.   |
| Response time is more.                                 | Response time is less.   |
| Not best for CRUD operations.                          | Best for CRUD operations.  |

## CONCEPTS UNDER REST:

**REST**- REST is an architecture used to create rest API (*To create Rest API we need a REST*).

**REST Assured**- To automate rest API we need rest assured libraries. (*For Ex. to automate web browser UI we need External JAR files in Selenium*)

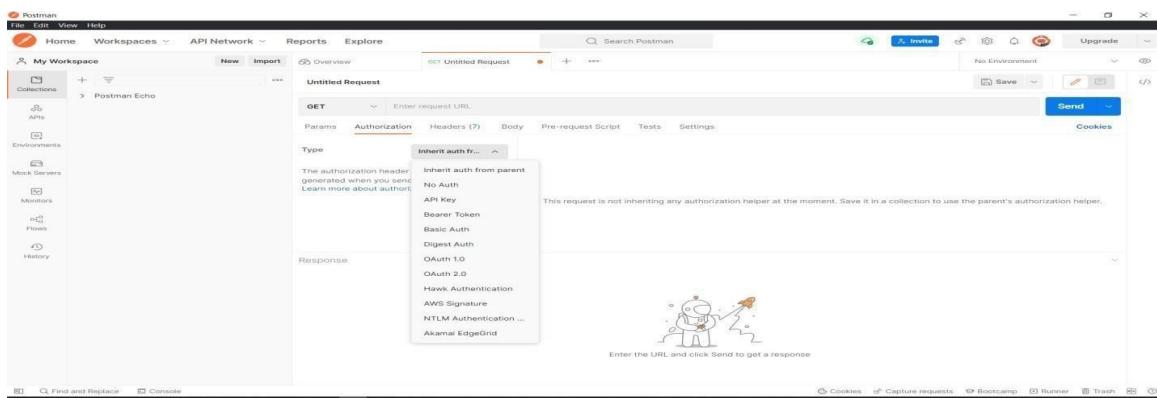
**RESTFUL**- when we automate rest API successfully it called as restful services.

## Webservices:

Whenever we are calling any API over http (internet) protocol it called as **Webservice**.

## DIFF. TYPES OF AUTHORIZATIONS-

- Basic Auth
- Digest
- Token
- OAuth1
- OAuth2
- No auth
- AWS signature



## 1) Basic Auth-

In Basic we have to pass **only Username and Password**.

The screenshot shows the Postman interface with a 'Basic Auth' request configuration. The 'Authorization' tab is selected. The 'Type' dropdown is set to 'Basic Auth'. The 'Username' and 'Password' fields are populated with redacted text. A note at the top right of the authorization section says: 'Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables.' A 'Send' button is visible at the top right of the request configuration area.

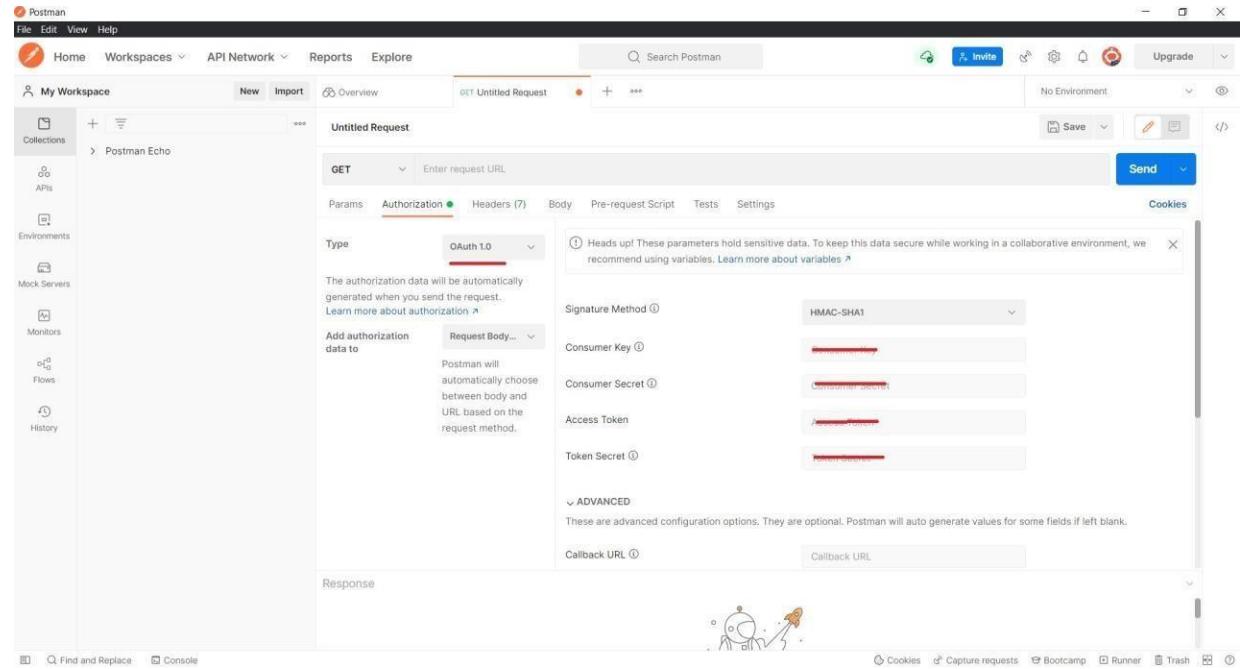
## 2) Digest Auth

In Digest auth we have to pass **Only Username and Password same** as that of Basic Auth but Digest Auth is more secure than Basic auth.

The screenshot shows the Postman interface with a 'Digest Auth' request configuration. The 'Authorization' tab is selected. The 'Type' dropdown is set to 'Digest Auth'. The 'Username' and 'Password' fields are populated with redacted text. A note at the top right of the authorization section says: 'Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables.' Below the basic fields, there is an 'ADVANCED' section with optional fields: 'Realm' (set to 'testrealm@example.com'), 'Nonce' (set to 'Nonce'), and 'Algorithm' (set to 'MD5'). A 'Yes, disable retrying the request' checkbox is also present. A note in the advanced section says: 'These are advanced configuration options. They are optional. Postman will auto generate values for some fields if left blank.' A 'Send' button is visible at the top right of the request configuration area.

### 3) OAuth 1.0

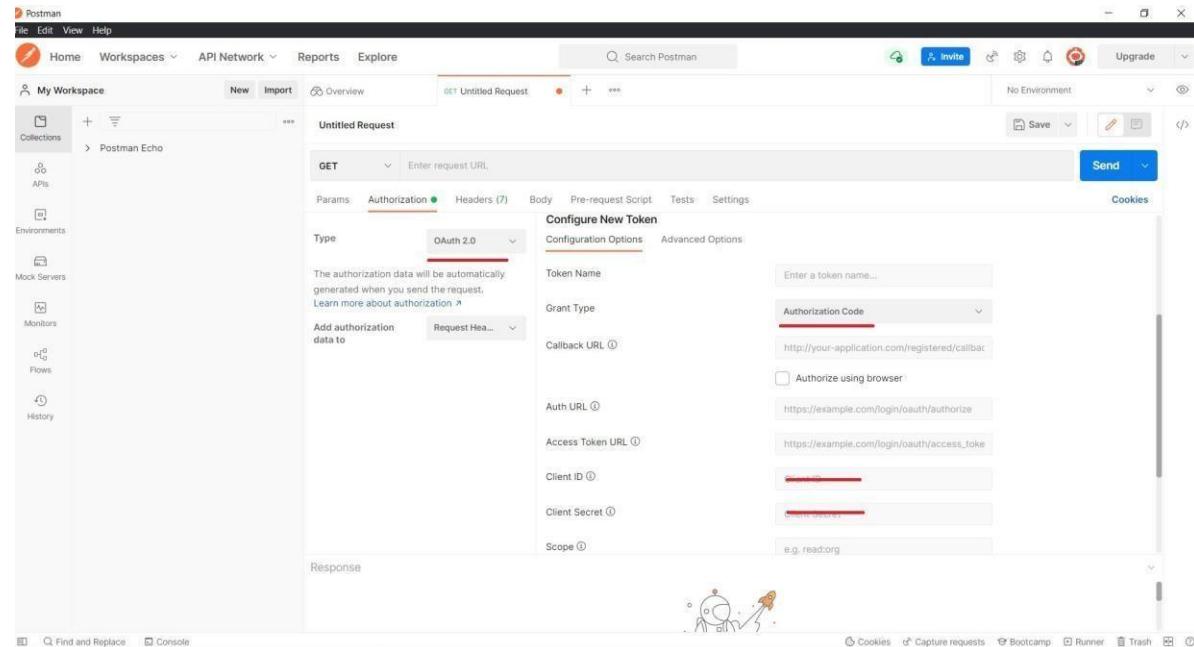
In OAuth 1.0 we have to pass have to pass Consumer Key, Consumers secrete, Access Token, Token Secrete. All this details will be provided by developer side.



The screenshot shows the Postman application interface. A 'GET Untitled Request' is selected. In the 'Authorization' tab, the 'Type' dropdown is set to 'OAuth 1.0'. The 'Signature Method' is set to 'HMAC-SHA1'. The 'Consumer Key' field contains a redacted value. The 'Consumer Secret' field contains a redacted value. The 'Access Token' field contains a redacted value. The 'Token Secret' field contains a redacted value. Below these fields, there is an 'ADVANCED' section with a 'Callback URL' field containing 'Callback URL'. The 'Response' panel is visible at the bottom.

### 4) OAuth 2.0

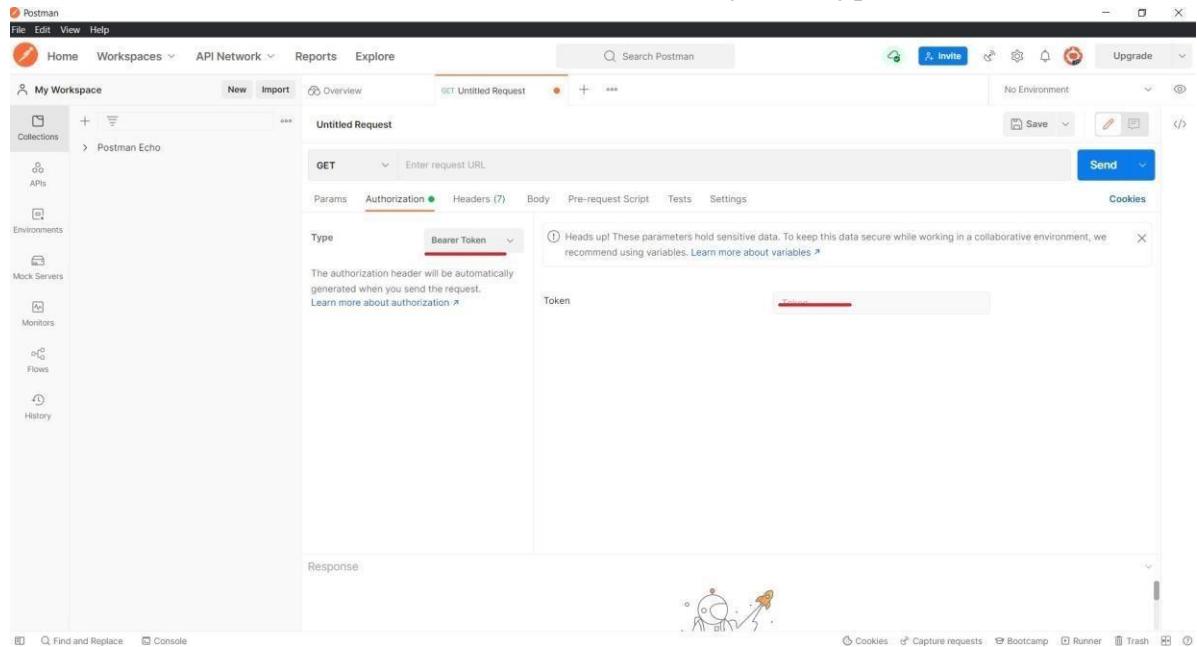
In OAuth 2.0 we have to pass have to pass Grant type, Client Id, Client Secrete. All this details will be provided by developer side.



The screenshot shows the Postman application interface. A 'GET Untitled Request' is selected. In the 'Authorization' tab, the 'Type' dropdown is set to 'OAuth 2.0'. The 'Configure New Token' section is expanded, showing 'Configuration Options' and 'Advanced Options'. Under 'Configuration Options', the 'Grant Type' is set to 'Authorization Code'. The 'Callback URL' field contains 'http://your-application.com/registered/callback'. The 'Auth URL' field contains 'https://example.com/login/oauth/authorize'. The 'Access Token URL' field contains 'https://example.com/login/oauth/access\_token'. The 'Client ID' field contains a redacted value. The 'Client Secret' field contains a redacted value. The 'Scope' field contains 'e.g. read:org'. The 'Response' panel is visible at the bottom.

## 5. Token-

Here we have to just put the value of **token**. Token value may be the combination of integer and character values. While entering the token value, we need to mention **Bearer** keyword. Bearer means the identification for that token. **Token is mostly used type of authorizations.**

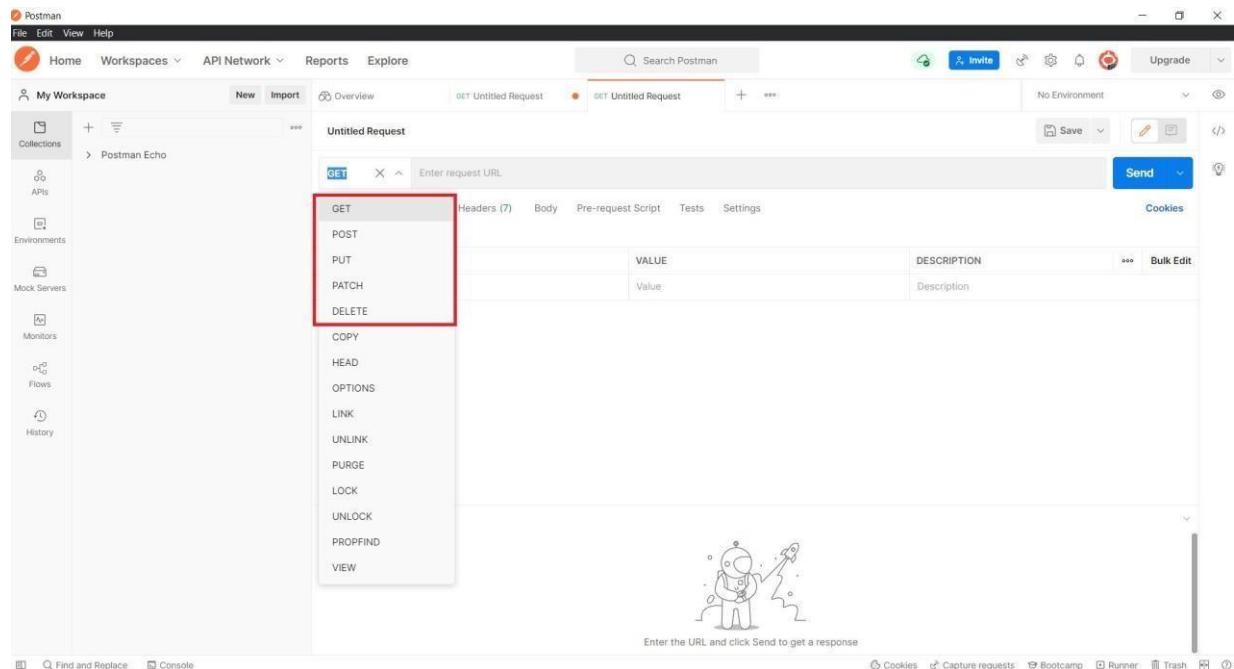


## CRUD OPERATIONS-

- C**-Create the Data- **POST**
- R**- Retrieve/ Fetch the Data-**GET**
- U**- Update the data- **PUT**
- D**- Delete the data- **DELETE**

**There are four different methods present in API which are-**

- 1) GET**- Used to fetch the data.
- 2) POST**- Used to create the data.
- 3) PUT**- Used to update the data.
- 4) DELETE**- Used to delete the data.



Types of status code:

- **1xx: Informational** – Communicates transfer protocol-level information.
- **2xx: Success** – Indicates that the client's request was accepted successfully.
- **3xx: Redirection** – Indicates that the client must take some additional action in order to complete their request.
- **4xx: Client Error** – This category of error status codes points the finger at clients.
- **5xx: Server Error** – The server takes responsibility for these error status codes.

**ERROR/STATUS codes-(there are many other status codes available on internet)**

- |                 |                              |
|-----------------|------------------------------|
| 1) 201- Created | when we create data          |
| 2) 200- OK      | when we get Successful data. |

- |                               |  |
|-------------------------------|--|
| 3) 400-Bad Request            | when URL wrong or end point missing.                           |
| 4) 401- Unauthorised          | when session got expired, passing invalid token/username/pass. |
| 5) 403- Forbidden             |  |
| 6) 404-Page not found.        | when we are trying to access the URL but URL not present       |
| 7) 500- Internal server Error | when any server down or network issue.                         |
| 8) 503-Service not available  |  |

## Test Environment-

The screenshot shows the Postman application interface. In the top right corner, there is a message box with the text "When we want to hit any API, We need to mention the Environment type" and a red box highlighting the "No Environment" button. The main workspace shows an "Untitled Request" with a GET method and an empty URL field. The left sidebar lists collections, APIs, environments, mock servers, monitors, flows, and history.

## Content-type-

The screenshot shows the Postman application interface. The "Headers" tab is selected in the request configuration. Under the "Content-Type" key, a dropdown menu is open, listing various media types: application/atom+xml, application/ecmascript, application/json, application/vnd.api+json, application/javascript, application/octet-stream, and application/ogg. The "application/json" option is highlighted. The main workspace shows an "Untitled Request" with a GET method and an empty URL field. The left sidebar lists collections, APIs, environments, mock servers, monitors, flows, and history.

## **Which type of Authorization is mostly used in organization? Why?**

**Token** Authorization is mostly used in any organization because if it expires then we can easily regenerate the new token by simple clicking on regenerate the token. Token expiry limit will be decided by Developer. Token also provides more security.

### **Theory regarding soapui**

#### **WSDL-**

WSDL stands for Web Service Descriptions Language. WSDL is basically an XML document contains all the details about web services & all API requests.

#### **UDDI-**

UDDI stands for Universal Description Discovery integration. UDDI is an XML based standard for describing, publishing and finding the web services.

#### **SOAP ELEMENTS-**

- Envelop- It is beginning and end of message.
- Header- Header elements contain header information.
- Body- Body element contains call & response information.
- Fault- Fault contains error and status information.

#### **WSDL ELEMENTS:**

- Type- Define the data types used by the web services
- Message – Define the data element for each operation
- Port Type- Describe the operation that can be performed and message involve
- Binding- Fault contains error and status information.

#### **WSDL ELEMENTS:**

- Type- Define the data types used by the web services.
- Message – Define the data element for each operation.
- Port Type- Describe the operation that can be performed and message involve.
- Binding- Defines the protocol and data format for each port type.

#### **Diff assertion**

content- to verify string

not content- same as above but opposite

valid http status code- to verify status code

invalid http status code- same as above but opposite

response SLA- to verify response time

see the below page

(scroll down)

## SOME INTERVIEW Q&A

### What are the different methods present in API?

There are different methods present in API:

GET

POST

PUT

DELETE

PATCH

### What are different operations performed in API?

Below are the operations performed in API:

GET- used to fetch the data

POST- Used to create data

PUT- Used to update data

DELETE- used to delete data

### What is difference between PUT and PATCH?

PUT- we can update all the fields as well as single field

PATCH- we can update single/ partial fields

### What are main differences between API and Web Service?

Api call internally and webservices call over the internet

The only **difference** is that a **Web service** facilitates interaction **between** two machines over network. An **API** acts as an interface **between** two **different** applications so that they can communicate with each other. **Web service** also uses SOAP, REST, and XML- RPC as a means of communication.

### What are the advantages of API Testing?

- a) Api provides the security.
- b) API checks the authentication and the data that we are passing.
- c) Can transfer the load to diff micro services.

- d) API helps to avoid data breaching.
- e) Test for Core Functionality.
- f) Time Effective- we can hit lots of APIs within less time.
- g) Language-Independent- like Json, XML, html, text.
- h) Easy Integration with GUI.

### **What is different Test Environment in project?**

- Generally we will have below four test Environments:
  - DEV- where developers works.
  - SIT/QA- where Testers works.
  - UAT- where Testers and Client works.
  - PROD- It's a live environment.

### **What are the test environments of API?**

Global- Global has large scope (used to pass variables between diff collections)

Local – Local has small scope (Used to pass variable from one request to another)

-we are using QA/UAT environment in which we are using Global and Local environment for API methods.

### **What must be checked when performing API testing?**

Error codes, data which are coming (Retrieval data), Time.

### **What are tools could be used for API testing?**

Postman

Swagger

SoapUI

Etc.

## **What are differences between API Testing and UI Testing?**

**API** doesn't provide the GUI ( Graphical User interface) but **UI** provides.

Types of status code:

- **1xx: Informational** – Communicates transfer protocol-level information.
- **2xx: Success** – Indicates that the client's request was accepted successfully.
- **3xx: Redirection** – Indicates that the client must take some additional action in order to complete their request.
- **4xx: Client Error** – This category of error status codes points the finger at clients.
- **5xx: Server Error** – The server takes responsibility for these error status codes.

## **What are common API errors that often founded?**

These are the common error getting during API testing

201-created

200-ok

400-Bad request

401-Unauthorised

403- Forbidden

404- Page not found

500- Internal server error

503-service not available

## **Any examples why error code generates?**

200- When we get successful data.

201- When we create data into database.

400- URL wrong or end point missing.

401- When session got expired, passing invalid token/ username/pass.

404- When we are trying to access the URL but URL not present.

405- Method not allowed.

500- Any server down or network issue.

**What are the responsibilities/ principles of API:**

- To write the test case and scenarios
- To execute the api's
- To validate the api's
- To open the defect if found any

**What are the collections?**

Collections are used to store the services (API methods)

By using collection we can run all the methods at the same time.

We can Import/Export Collection.

### **What is mean bearer token?**

Bearer token is one of the Authentication pass in headers

Bearer means identification for the token.

### **Where we pass the data in post?**

We pass the data in Body-> Raw-> in the form of Json, XML, Html, text

### **Can we run collection?**

Yes, we can run the collection and collection methods at the same time, but before we run the previous or old collection we have to update the authentication.

### **What is mean by end points/service URL?**

End points are the different service URLs which are used to hit the URL with domain URL.

### **What is mean API?-Application programming interface**

API stands for Application programming interface.

- Used to communicate between two systems.
- It simply knows as sending the request and getting the response.

### **What are headers?**

Headers is nothing but the what kind of request it is

```
{content-type= application json/  
application xml/  
application text }
```

## **What is bearer?**

Bearer is the identifier for particular token used for the Authentication.

## **Difference between SOAP and REST**

| <b>SOAP</b>  | <b>REST</b>  |
|--|--|
| SOAP is Protocol.                                      | REST is Architecture.  |
| Uses XML only.   | Uses XML, JSON, HTML, TEXT, JavaScript etc                           |
| SOAP reads WSDL file.                                  | REST reads API only.   |
| Type of security provided by SOAP is SOAP Environment. | Type of security provided by REST is AUTH Token, HEADER, PARAMS etc. |
| Heavy in weight API.                                   | Light in weight API.   |
| Response time is more.                                 | Response time is less.   |
| Not best for CRUD operations.                          | Best for CRUD operations.  |

## **Types of API**

REST API- Uses Postman tool (Representational state transfer)

SOAP API- Uses SOAPUI tool (simple object access protocol)

## **Concept under REST:**

REST- REST is an architecture used to create rest API.

REST Assured- To automate rest API we need rest assured libraries.

RESTFUL- when we automate rest API it called as restful services.

## **What is the difference between '/' and '?'**

/- Path parameter

Ex: <https://gorest.co.in/public/v2/users>

?- Query parameter

Ex: <https://gorest.co.in/public/v2/users?name=madhuri>

## **What is producer and consumer?**

Producer- who produce the data

Consumer- who consumes the data

## **What is URI?**

URI- Unique resource identifier

URI= URL+ENDPOINT

Eg. <https://www.amazon.com/>login/home

## **2. What are diff ways to pass the data/ scripting languages?**

### **a) JSON:**

```
{  
  "name": "Suraj ",  
  "email": "Suraj123@gmail.com",  
  "gender": "Male",  
  "status": "Active"  
}
```

### **b) XML:**

```
<name>suraj</name>  
<email>suraj@gmail.com</email>
```

### **c) String**

### **d) Text**

### **e) Html**

### **f) Javascript**

Etc.

## **What are headers?**

Headers mean what kind of data we are passing.

- a. Authorization
- b. Content Type
- c. Language
- d. Etc.

## **What we pass in http request?**

- a. URI
- b. Headers
- c. Payload

## **Diff between Api and webservice**

### **Webservice**

Webservices hit over the internet

A web service require a Network for it's operation

All web services are APIs

Webservices hits Online

Web service is used to communicate between two machines over a network

it works only for web applications  
ex: amazon,gpay

### **Api**

Api hit over the internet as well as internally

API service may or may not require a Network for it's operation

Not all APIs are web services

Api hits online as well as offline

An API used as an interface between two different applications for communicating with each other

it works for web as well as desktop applications  
ex: amazon, word, paint

## **What are different authorizations?**

- a. Basic Auth

pass the username and pass.

b. Digest

Whenever we are passing username and pass it will get convert in # keys.

It means your username/pass will secured get server side too.

c. Oauth1

Oauth1 required below things:

1. Consumer Key
2. Consumer Secret
3. Access Token
4. Secret Token

*Above info will get from developers*

e. Oauth2

Oauth2 required below things:

1. Client Id
2. Client Secret
3. Grant type

*Above info will get from developers.*

f. Bearer Token

g. NoAuth

## **What are Oauth1 and Oauth2?**

Oauth1- this auth uses when we need third party logins.

Oauth2- this auth uses when we have single url and different endpoints

## **What are different API gateways.**

- a. SSL certificate
- b. Routing
- c. Adapter
- d. Cache

e. Load balancer

### **Difference between monolithic and microservice?**

Monolithic - all api available under one service

Microservice- for api have different microservice.

### **What is means URI(API)**

URI= URL+endpoints(resource)

url: [www.facebook.com](http://www.facebook.com)

endpoints: /login/home

URI- unified resource identifier

url- unified resource locator

### **What are diff validations points in api:**

- Data from the response
- status code
- response time

## **Theory related to soapUI**

### **What is WSDL file**

WSDL basically an XML document contains all the details about web service and all API request

### **What is Web service?**

Whenever we are hitting any service over the internet it known as webservice.

Webservice is any piece of software that makes it available over the internet and uses a standardized XML messaging system.

### **What is UDDI?**

- Universal description discovery integration
- UDDI is an XML based standard for describing, publishing and finding the webservices.

### **What are diff soap elements/components?**

- a. Envelop – It is beginning and end of message.
- b. Header – Header elements contain header information.
- c. Body – body element contains call and response information.
- d. Fault – Fault contain error and status information.

### **What is diff WSDL element/component?**

- e. Type- Define the data types used by the webservices.
- f. Message – Define the data element for each operation.
- g. Port Type- Describe the operation that can be performed and message involve.
- h. Binding- Defines the protocol and data format for each port type.

### **What are different assertions present in SOAPUI?**

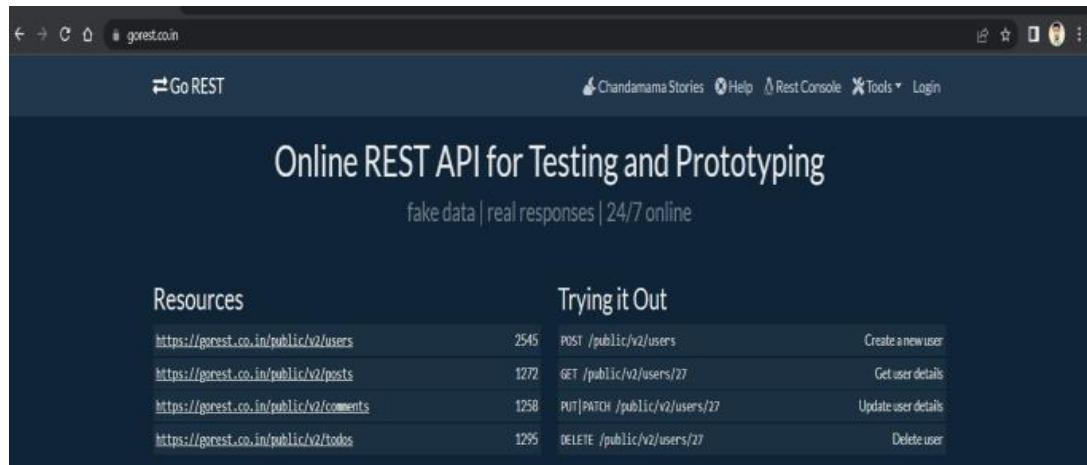
Below are the different assertions present in SOAPUI:

- a. Contains - checks for the existence of a specified string
- b. Not Contains - checks for the non-existence of a specified string
- c. Response SLA - validates that the last received response time was within the defined limit. Applicable to Script TestSteps and TestSteps that send requests and receive responses.
- d. Invalid HTTP Status Codes - checks that the target TestStep received an HTTP result with a status code not in the list of defined codes. Applicable to any TestStep that receives HTTP messages
- e. Valid HTTP Status Codes - checks that the target TestStep received an HTTP result with a status code in the list of defined codes. Applicable to any TestStep that receives HTTP message

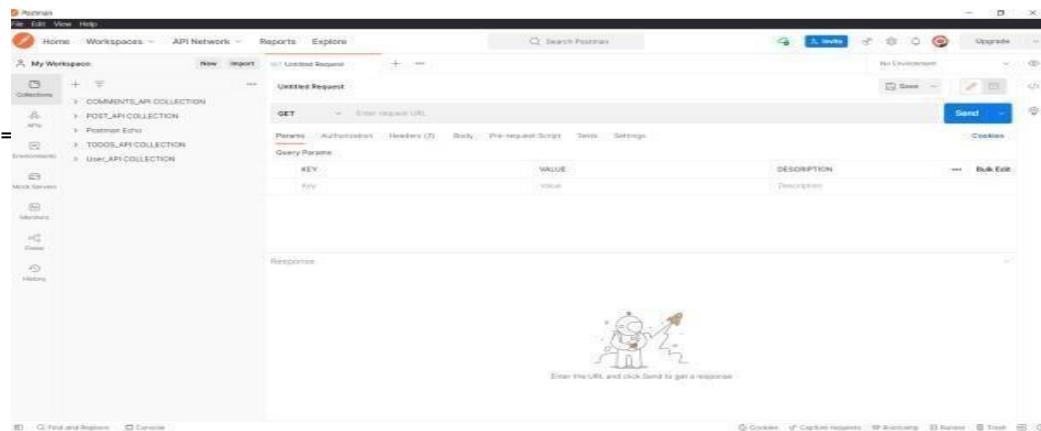
## HOW TO hit services IN POSTMAN-

### Steps:

1. Open the postman as well as Gorest site.



The screenshot shows the Gorest API homepage. At the top, there's a navigation bar with links for 'Go REST', 'Chandamama Stories', 'Help', 'Rest Console', 'Tools', and 'Login'. Below the navigation, the text 'Online REST API for Testing and Prototyping' is displayed, followed by 'fake data | real responses | 24/7 online'. On the left, a sidebar titled 'Resources' lists four API endpoints: 'https://gorest.co.in/public/v2/users', 'https://gorest.co.in/public/v2/posts', 'https://gorest.co.in/public/v2/comments', and 'https://gorest.co.in/public/v2/todos'. To the right, a section titled 'Trying it Out' shows details for each endpoint: '2545 POST /public/v2/users' (Create a new user), '1272 GET /public/v2/users/27' (Get user details), '1258 PUT/PATCH /public/v2/users/27' (Update user details), and '1295 DELETE /public/v2/users/27' (Delete user).



The screenshot shows the Postman application interface. The top navigation bar includes 'File', 'Edit', 'View', 'Help', 'Home', 'Workspaces', 'API Network', 'Reports', and 'Explore'. The main workspace on the left is titled 'My Workspace' and contains several collections: 'COMMUNICATE-API COLLECTION', 'POST\_API\_COLLECTION', 'Position Env.', 'TODOES\_API\_COLLECTION', and 'USER\_API\_COLLECTION'. In the center, a 'GET' request is being configured for the 'User API Collection'. The 'Headers' tab is selected, showing a single header 'Content-Type: application/json'. Below the request configuration, there's a 'Response' section with a cartoon character and the instruction 'Enter the URL and click Send to get a response!'. The bottom of the screen shows a footer with various icons and links.

2. Select the method (i.e GET, PUT, POST, and DELETE).

The screenshot shows the Postman application interface. On the left, there's a sidebar with sections like 'My Workspace' (Collections, APIs, Environments, Mock Servers, Monitors, Flows, History), 'Find and Replace', and 'Console'. The main area is titled 'Untitled Request' with tabs for 'Headers (7)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. A dropdown menu is open over the 'Method' field, which is currently set to 'GET'. The dropdown options include: GET, POST, PUT, PATCH, DELETE, COPY, HEAD, OPTIONS, LINK, UNLINK, PURGE, LOCK, UNLOCK, PROPFIND, and VIEW. The 'Send' button is visible at the top right of the main panel.

3. Add the URL/API.

4. Add the headers

- a. Authorization
- b. Content-type

5. Only in case of POST and PUT add the body not for the GET.

6. Click in send.

**HOW TO GET(FETCH) THE RESPONSE BY HITTING API IN POSTMAN-**

## Step1

The Postman interface shows the 'User\_API COLLECTION / User get' endpoint selected. The 'Authorization' tab is active, showing a 'Bearer Token' type with a placeholder 'Token' and a 'Add Token' button. A note about sensitive data and variables is visible.

## Step2

The 'Headers' tab is selected, showing a table with a single row for 'Content-Type' set to 'application/json'. A note at the top right says 'Hit the API by clicking on Send'.

## Step3

The 'Body' tab is selected, displaying a JSON response object. The status bar at the bottom indicates 'Status: 200 OK', 'Time: 64 ms', and 'Size: 3.14 KB'. The 'Status code' field is also shown.

```
23: { "id": 2816,
24:   "name": "Apsara Shazma",
25:   "email": "apsara_shazma@towne.name",
26:   "gender": "male",
27:   "status": "active"
28: }
29:
```

## HOW TO POST(CREATE) THE RESPONSE BY HITTING API IN POSTMAN-

### Step1

The screenshot shows the Postman interface with the 'User API Collection / User post' endpoint selected. In the 'Headers' tab, there is an 'Authorization' field set to 'Bearer Token' with a placeholder value 'faa5f35da070ba9f0fc2a271bf97f14cd5a82...'. A note below the field says: 'The authorization header will be automatically generated when you send the request. Learn more about authorization'.

### Step2

The screenshot shows the Postman interface with the 'User API Collection / User post' endpoint selected. In the 'Headers' tab, a new 'Content-Type' header is added with the value 'application/json'. A note below the table says: 'In Headers Add content type'.

### Step3

The screenshot shows the Postman interface with the 'User API Collection / User post' endpoint selected. In the 'Body' tab, a JSON payload is being entered:

```
1 {  
2   "name": "Anvesh Sharma",  
3   "email": "sharma.anvesh@gmail.com",  
4   "gender": "male",  
5   "status": "inactive"  
6 }  
7
```

A red box highlights the 'Send' button. Below the body, a note says: 'Provide body and details which you want to create'. The status bar at the bottom right shows: Status: 201 Created Time: 88 ms Size: 854 B Save Response.

## HOW TO PUT(UPDATE) THE RESPONSE BY HITTING API IN POSTMAN-

## Step1

The Postman interface shows a collection named 'User\_API COLLECTION / User put'. A PUT request is selected with the URL <https://gorest.co.in/public/v2/users/2811>. The 'Authorization' tab is active, showing 'Bearer Token' selected. A placeholder text 'Enter API with Id for which you want to update the Data' is present. The token 'faa5f35da070ba9f0fc2a2718f97f14cd5a82...' is entered in the token field.

## Step2

The Postman interface shows the same collection and request setup as Step 1. The 'Headers' tab is now active, showing 'Content-Type: application/json' selected. The 'Send' button is visible at the top right.

## Step3

The Postman interface shows the collection and request setup. The 'Body' tab is active, showing 'raw' selected. A JSON object is defined with fields: name, email, gender, status, and id. The value for 'name' is 'Chandrabhan Acharya'. The 'Send' button is highlighted with a red border. Below the request area, the response status is shown as 'Status: 200 OK'.

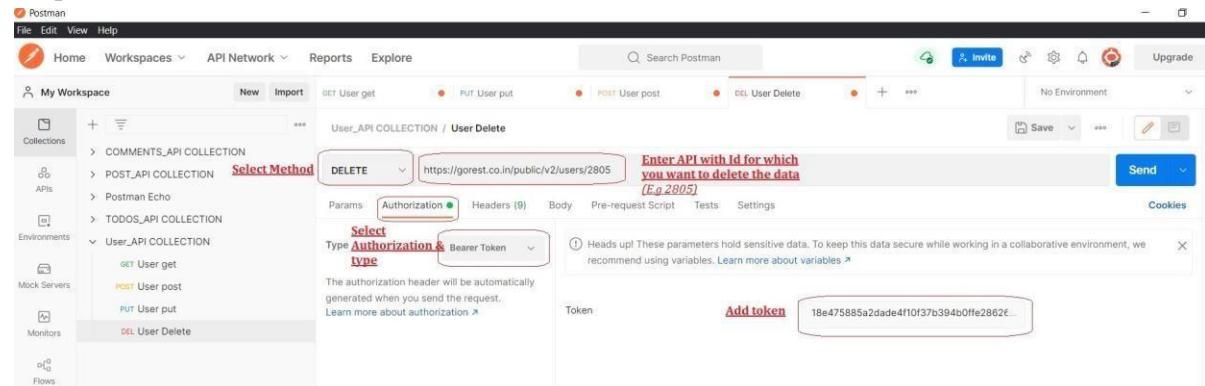
```
1 {  
2   "name": "Chandrabhan Acharya",  
3   "email": "acharya_chandrabhan@sawayn-jones.name",  
4   "gender": "male",  
5   "status": "active",  
6   "id": 2811  
7 }
```

A note 'Provide body with details to be updated' is displayed next to the body input field. The response body is shown as 'Updated Response with Id'.

- ✓ Get token from gorest site(login->in howdy-Api token)
- ✓ Ex: AUTHORIZATION- Bearer kdhfkgsfk745874697456740bhchjfvx4889

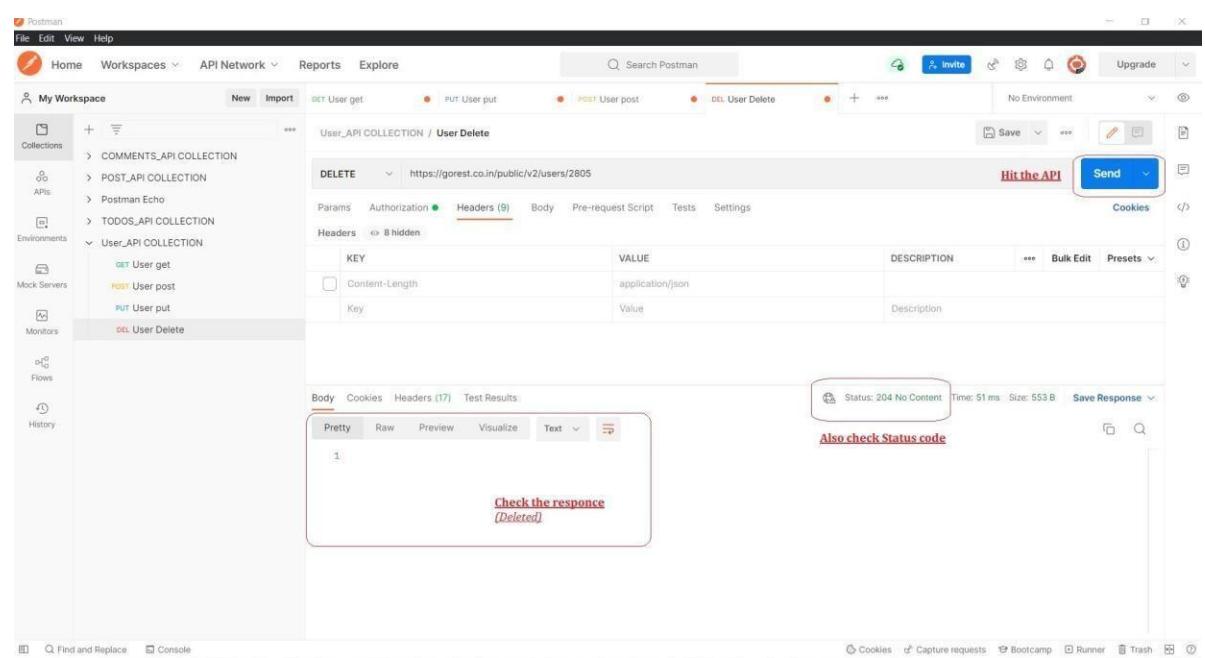
## HOW TO DELETE THE RESPONSE BY HITTING API IN POSTMAN-

### Step1



The screenshot shows the Postman interface with the 'User\_Delete' API selected in the collection tree. The main request details are displayed: Method: DELETE, URL: https://gorest.co.in/public/v2/users/2805, and Headers tab selected. A red box highlights the 'Select Method' dropdown and the URL field. A red text overlay says 'Enter API with Id for which you want to delete the data (E.g 2805)'. The 'Authorization' dropdown is also highlighted.

Ste



The screenshot shows the Postman interface after sending the DELETE request. The 'Send' button is highlighted with a red box. The response status is shown as 204 No Content. A red text overlay says 'Also check Status code'. Below the status, a red box highlights the 'Check the response (Deleted)' message in the response body.

How to hit services in SOAPUI

# 1. How to hit WSDL services

Steps in Soap service:

1. click on soap project(wsdl file):

<http://www.dneonline.com/calculator.asmx?WSDL>

2. ADD wsdl file/url and give project name, make all

checkboxes as bydefault

eg. <http://www.dneonline.com/calculator.asmx?WSDL>

3. click on any api (+) and then click on request

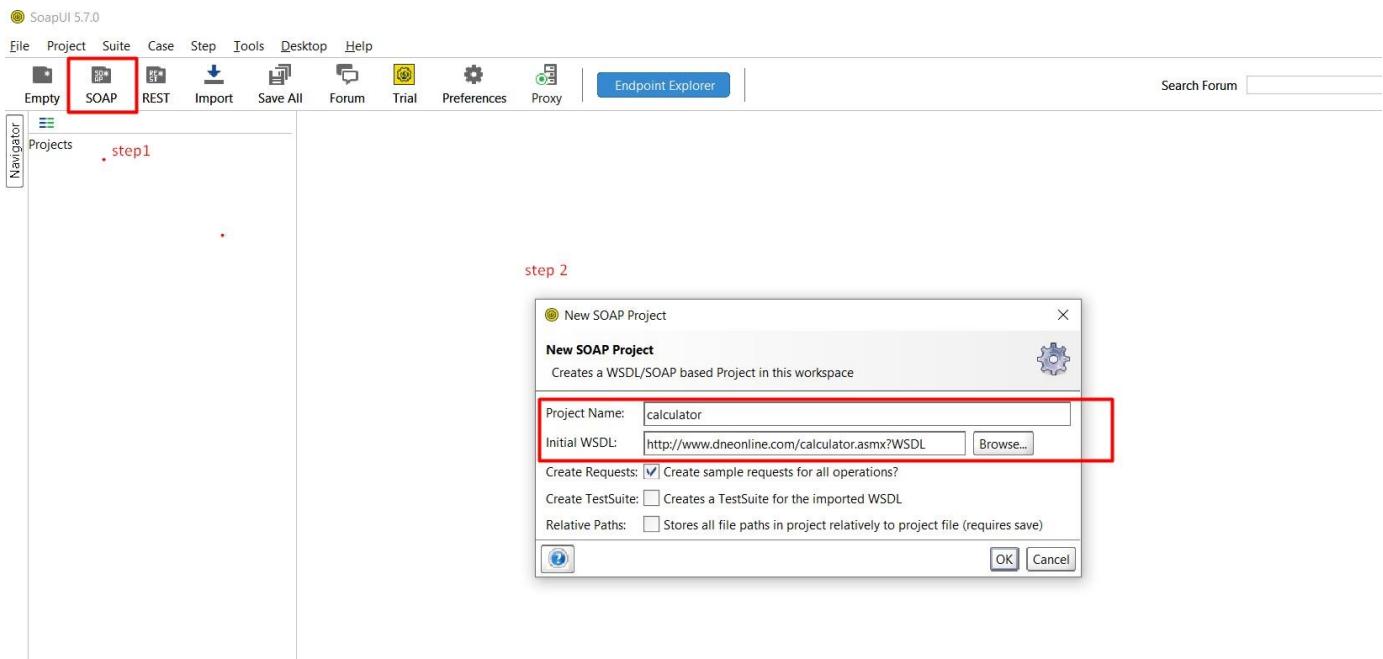
4. in request pass then values

eg. in add pass 10 and 10 as int a and int b by removing

question mark

5. click on green run button.

6.now you will get the response.



SoapUI 5.7.0

The screenshot shows the SoapUI interface with the following details:

- Navigator:** Projects > calculator > Request 1.
- Request 1:** URL: http://www.dneonline.com/calculator.asmx
  - Raw XML:** Shows the SOAP request XML with parameters tem:intA=5 and tem:intB=5.
  - Response:** Shows the SOAP response XML with the result AddResult=10.
- Request Properties:** Name: Request 1, Value: 291.

SoapUI 5.7.0

The screenshot shows the SoapUI interface with the following details:

- Navigator:** Projects > calculator > Request 1.
- Request 1:** URL: http://www.dneonline.com/calculator.asmx
  - Raw XML:** Shows the SOAP request XML with parameters tem:intA=5 and tem:intB=5.
  - Response:** Shows the SOAP response XML with the result AddResult=10.
- Request Properties:** Name: Request 1, Value: 291.

Same for the divide, multiply, substract

You can apply the assertion for rest as well as for soap(wsdl)

service

Note: assertions are shown in rest service in below

## 2. How to hit rest services in soapui tool

Steps in rest service:

1. click on rest project

2. pass the url from service

eg. <https://gorest.co.in/public/v2/users>

3. add the method(GET)

4. by clicking on + button add headers in header

key:Authorization

key:Content-type

note: make sure all spells are correct

5. you will get response here but you can't apply any assertions.

6. to apply any assertion/validations right click on project and select-

> new test suit

- give any name for that test suit

7. now right click on testsuit and create a new test case-> new test

case

8. then right click on testcase and add new test request->add step->

rest request

9. select the request(automatically populate the request)

10. now you will get the assertion tab below side

11. click on assertion and then click on + icon

12. you will get the diff assertion

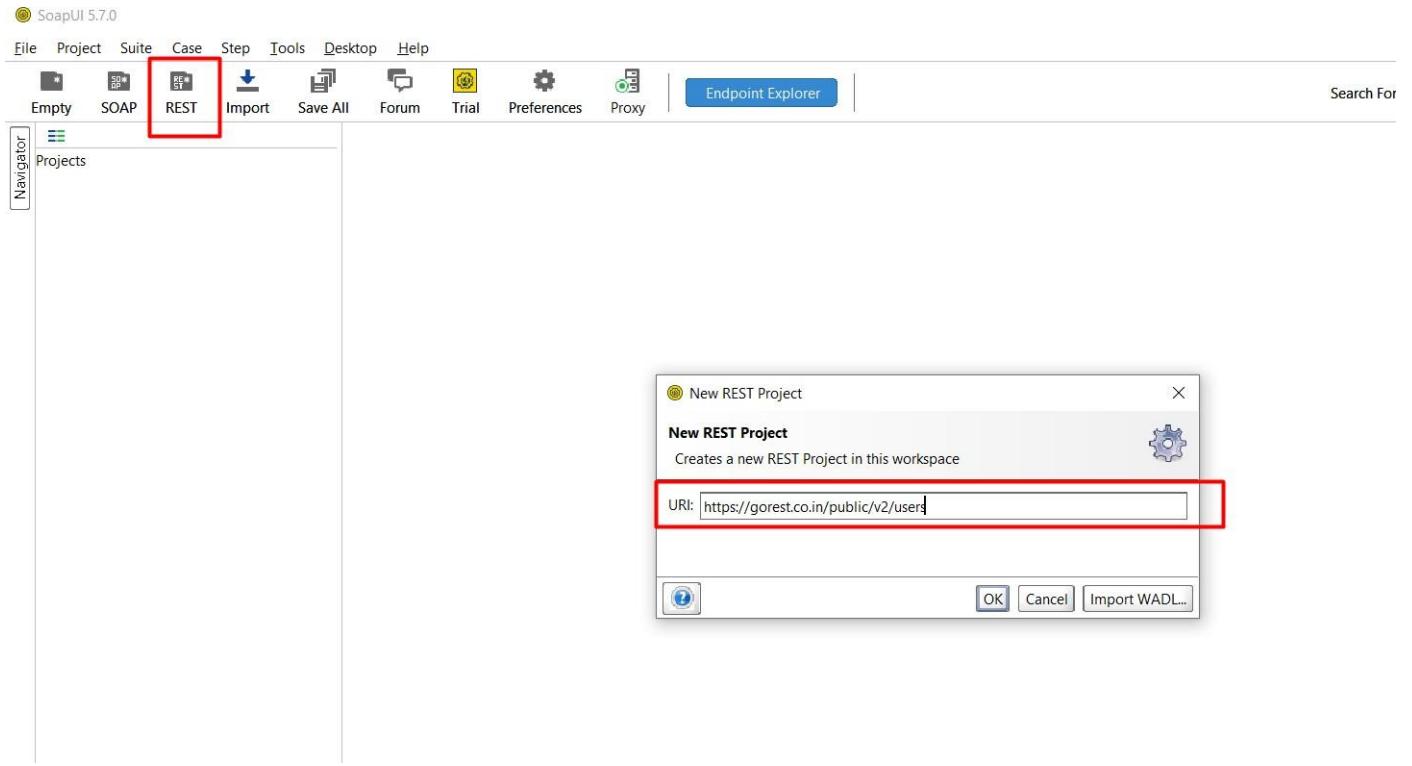
eg. content- to verify string

not content- same as above but opposite

valid http status code- to verify status code

invalid http status code- same as above but opposite

response SLA- to verify response time



The screenshot shows the SoapUI interface with the REST tab selected. It displays a REST project structure and a request configuration. The request details are as follows:

- Method:** GET
- Endpoint:** https://gorest.co.in /public/v2/users
- Parameters:** ?Authorization=Bearer 6e1c6fb50d56d2e8c2f51f23844b4445fa7640c6566549395243e9493f60a7d4&Content-Type=application/json
- Headers:**
  - Authorization: Bearer 6e1c6fb50d56d2e8.QUERY
  - Content-Type: application/json

The response content is a JSON array of user objects:

```

[{"id": 2590, "name": "Rohit Nehru", "email": "rohit_nehru@mavert.org", "gender": "female", "status": "active"}, {"id": 2587, "name": "Datta Sethi", "email": "datta_sethi@hilll.name", "gender": "female", "status": "inactive"}, {"id": 2585, "name": "Gajaadhar Shah", "email": "gajaadhar_shah@connelly.org", "gender": "male", "status": "active"}, {"id": 2584, "name": "Gudakesa Bandopadhyay", "email": "bandopadhyay_gudakesa@collins.biz", "gender": "male", "status": "active"}, {"id": 2583, "name": "Kiran Kumar", "email": "kiran_kumar@soliton.com", "gender": "male", "status": "active"}]
  
```

To apply the assertions need to convert the project into test cases

Empty SOAP REST Import Save All Forum Trial Preferences Proxy Endpoint Explorer

Projects step 1

REST Project 1

Method: Enter

Endpoint: https://gorest.co.in

Resource: /public/v2/users

Parameters: ?Authorization=Bearer 6e1c6fb50d56d2e8

| Value                         | Style | Le       |
|-------------------------------|-------|----------|
| Bearer 6e1c6fb50d56d2e8.QUERY |       | RESOURCE |
| application/json              | QUERY | RESOURCE |

Add WSDL Ctrl-U  
Add WADL Ctrl-F  
Import Swagger/OpenAPI Definition  
Export Swagger/OpenAPI Definition  
Import From SwaggerHub

New REST Service from URI

Launch TestRunner  
Launch LoadTestRunner  
Launch HTTP Monitor  
Launch Security TestRunner

New TestSuite Ctrl-T  
New SOAP MockService Ctrl-O  
New REST MockService Ctrl-R

Rename F2  
Remove Delete  
Reload Project F5  
Resolve  
Close Project

Save Project Ctrl-S  
Save Project As Ctrl+Shift-S

Sets if parameter is required

Add.. Edit.. Remove

Empty SOAP REST Import Save All Forum Trial Preferences Proxy Endp

Navigator

Projects

REST Project 1

- https://gorest.co.in
- Users [/public/v2/users]
- GET Users 1
- Request 1

TestSuite 1

step 1

Request TestCases

Show TestSuite Editor Enter

Disable TestSuite

New TestCase Ctrl-N step2

Clone TestSuite F9

Launch TestRunner

Rename F2

Remove Delete

Move TestSuite Up Ctrl-Up

Move TestSuite Down Ctrl-Down

Export

Import Test Case

Description Properties Setup Script TearDown

TestSuite Log

The screenshot shows the SoapUI interface with a context menu open over a 'TestSuite 1' item. The menu includes options like 'Show TestSuite Editor', 'Disable TestSuite', and several creation and modification options. The 'New TestCase' option is highlighted with a red box and labeled 'step2'.

## SoapUI 5.7.0

File Project Suite Case Step Tools Desktop Help

Empty SOAP REST Import Save All Forum Trial Preferences Proxy Endpoint Explorer

**Projects**

- REST Project 1
  - https://gorest.co.in
    - Users [/public/v2/users]
      - GET Users 1
        - Request 1
  - TestSuite 1
    - TestCase 1

**Request 1**

TestSuite 1

TestCase 1

Raw Request Test On Demand

Add Step

- SOAP Request
- REST Request**
- HTTP Request
- AMF Request
- JDBC Request
- GraphQL Request
- Properties
- Property Transfer
- Conditional Goto
- Run TestCase
- Groovy Script
- Delay
- SOAP Mock Response
- Manual TestStep
- Receive MQTT Message
- Publish using MQTT
- Drop MQTT Connection

TestCase Properties Custom Properties

| Property | Value      |
|----------|------------|
| Name     | TestCase 1 |

Auth Headers (0) Attachments (0) Representations (0) JMS Headers JMS Property (0)

response time: 168ms (2254 bytes)

File Project Suite Case Step Tools Desktop Help

Empty SOAP REST Import Save All Forum Trial Preferences Proxy Endpoint Explorer Search Forum Online Help

**Projects**

- REST Project 1
  - step 1
    - Endpoint https://gorest.co.in
  - Users [/public/v2/users]
    - Users 1
      - Request 1
  - TestSuite 1
    - TestCase 1
      - Test Steps (1)
        - REST Request

**Request**

Resource/Method: https://gorest.co.in/public/v2/users

| Name          | Value | Style    | Level |
|---------------|-------|----------|-------|
| Authorization | QUERY | RESOURCE |       |
| Content-Type  | QUERY | RESOURCE |       |

Raw XML JSON

Headers (26) Attachments (0) SSL Info (3 certs) Representations (1) Schema (conflicts) JMS (0)

Custom Properties REST TestRequest Properties

| Property    | Value        |
|-------------|--------------|
| Name        | REST Request |
| Description |              |
| Encoding    | UTF-8        |

Assertions (0) Request Log (1)

Response time: 168ms (2254 bytes)

Properties

SoapUI 5.7.0

File Project Suite Case Step Tools Desktop Help

Empty SOAP REST Import Save All Forum Trial Preferences Proxy Endpoint Explorer Search Forum Online Help

**REST Request**

Endpoint: https://gorest.co.in

Resource/Method: GET /Users

**Add Assertion**

Name:

Assertions

**Recently used**

- Property Content
- Compliance, Status and Standards
- Script
- SLA
- JMS
- JDBC
- Security

**Contains**

Searches for the existence of a string token in the property value, supports regular expressions. Applicable to any property.

**Invalid HTTP Status Codes**

Checks that the target TestStep received an HTTP result with a status code not in the list of defined codes. Applicable to any TestStep that receives HTTP messages.

**Not Contains**

Searches for the non-existence of a string token in the property value, supports regular expressions. Applicable to any property.

**Response SLA**

Validates that the last received response time was within the defined limit. Applicable to Script TestSteps and TestSteps that send requests and receive responses.

**Valid HTTP Status Codes**

Checks that the target TestStep received an HTTP result with a status code in the list of defined codes. Applicable to any TestStep that receives HTTP messages.

Auth Headers (0) Attachments (0)

Custom Properties

| Property    | Value        |
|-------------|--------------|
| Name        | REST Request |
| Description | UTF-8        |

Assertions (0) Request Log (1)

response time: 152ms (2268 bytes)

You will get the assertion anywhere in assertion popup.(just find the

highlighted assertions)

SoapUI 5.7.0

File Project Suite Case Step Tools Desktop Help

Empty SOAP REST Import Save All Forum Trial Preferences Proxy Endpoint Explorer Search Forum Online Help

**REST Request**

Endpoint: https://gorest.co.in

Resource/Method: GET /Users -> Users 1

**Contains Assertion**

Content: Datta

Ignore Case:  Ignore case in comparison

Regular Expression:  Use token as Regular Expression

OK Cancel

Raw XML

```

1 [
2   {
3     "id": 2587,
4     "name": "Datta Sethi",
5     "email": "datta_sethi@hilll.name",
6     "gender": "female",
7     "status": "inactive"
8   },
9   {
10     "id": 2585,
11     "name": "Gajaadhar Shah",
12     "email": "gajaadhar_shah@connelly.org",
13     "gender": "male",
14     "status": "active"
15   },
16   {
17     "id": 2584,
18     "name": "Gudakesha Bandopadhyay",
19     "email": "bandopadhyay_gudakesha@collins.biz",
20     "gender": "male",
21     "status": "active"
22   },
23   {
24     "id": 2583,
25     "name": "Prof. Anjaneya Chopra",
26     "email": "prof_anjaneya_chopra@ebert-jaskolski.biz",
27     "gender": "female",
28     "status": "active"
29   },
30 ]

```

Raw JSON

```

1 [
2   {
3     "id": 2587,
4     "name": "Datta Sethi",
5     "email": "datta_sethi@hilll.name",
6     "gender": "female",
7     "status": "inactive"
8   },
9   {
10     "id": 2585,
11     "name": "Gajaadhar Shah",
12     "email": "gajaadhar_shah@connelly.org",
13     "gender": "male",
14     "status": "active"
15   },
16   {
17     "id": 2584,
18     "name": "Gudakesha Bandopadhyay",
19     "email": "bandopadhyay_gudakesha@collins.biz",
20     "gender": "male",
21     "status": "active"
22   },
23   {
24     "id": 2583,
25     "name": "Prof. Anjaneya Chopra",
26     "email": "prof_anjaneya_chopra@ebert-jaskolski.biz",
27     "gender": "female",
28     "status": "active"
29   },
30 ]

```

Custom Properties

| Property    | Value        |
|-------------|--------------|
| Name        | REST Request |
| Description | UTF-8        |

Auth Headers (0) Attachments (0) Representations (0) JMS Headers JMS Property (0)

Assertions (1) Request Log (1)

response time: 152ms (2268 bytes)

SoapUI 5.7.0

File Project Suite Case Step Tools Desktop Help

The screenshot shows the SoapUI interface with a REST Project named "REST Project 1" containing an endpoint "https://gorest.co.in". A "REST Request" step is selected under "Test Steps (1)". The "Request" tab displays the following configuration:

| Name          | Value | Style    | Level |
|---------------|-------|----------|-------|
| Authorization | QUERY | RESOURCE |       |
| Content-Type  | QUERY | RESOURCE |       |

The "Response" tab shows the raw JSON response from the API call:

```
1 [
2   {
3     "id": 2587,
4     "name": "Datta Sethi",
5     "email": "datta_sethi@hilll.name",
6     "gender": "female",
7     "status": "inactive"
8   },
9   {
10    "id": 2585,
11    "name": "Gajadhar Shah",
12    "email": "gajadhar_shah@connelly.org",
13    "gender": "male",
14    "status": "active"
15  },
16  {
17    "id": 2584
18 }
```

Below the response, there is a red box highlighting the status message "Contains - VALID" and a tooltip "Assertions for this request".

Same for the others

SoapUI 5.7.0

File Project Suite Case Step Tools Desktop Help

The screenshot shows the SoapUI interface with the same project structure. A "REST Request" step is selected. A modal dialog box titled "Add Assertion" is open, showing various assertion types:

- Recently used**: Property Content, Compliance, Status and Standards, Script, SLA, JMS, JDBC, Security.
- Property Content**: Searches for the existence of a string token in the property value, supports regular expressions. Applicable to any property.
- Compliance, Status and Standards**: Invalid HTTP Status Codes, Response SLA, Response Status Codes.
- Script**: Not Contains, Response SLA, Response Status Codes.
- SLA**: Response SLA.
- JMS**: Response Status Codes.
- JDBC**: Response Status Codes.
- Security**: Response SLA, Response Status Codes.

A red box highlights the "Not Contains" option under the "Script" section. The "Add" button at the bottom right of the dialog is also highlighted with a red box.

SoapUI 5.7.0

File Project Suite Case Step Tools Desktop Help

Empty SOAP REST Import Save All Forum Trial Preferences Proxy Endpoint Explorer Search Forum Online Help

**REST Request**

Endpoint https://gorest.co.in

Resource/Method: GET Users -> Users 1 [/public/v2/users]

Request

| Name          | Value | Style    | Level |
|---------------|-------|----------|-------|
| Authorization | QUERY | RESOURCE |       |

NotContains Assertion

Specify options

Content: dicidscwefwiefkjndfsifkwl

Ignore Case:  Ignore case in comparison

Regular Expression:  Use token as Regular Expression

Auth

OK Cancel

Raw [HTML] JSON XML

```

1 [
2   {
3     "id": 2587,
4     "name": "Datta Sethi",
5     "email": "datta_sethi@hilll.name",
6     "gender": "female",
7     "status": "inactive"
8   },
9   {
10    "id": 2585,
11    "name": "Gajaadhar Shah",
12    "email": "gajaadhar_shah@connelly.org",
13    "gender": "male",
14    "status": "active"
15  },
16  {
17    "id": 2584
18  }
19 ]

```

Headers (26) Attachments (0) SSL Info (3 certs) Representations (1) Schema (conflicts) JMS

Online Help

Contains - VALID  
Not Contains - VALID

SoapUI 5.7.0

File Project Suite Case Step Tools Desktop Help

Empty SOAP REST Import Save All Forum Trial Preferences Proxy Endpoint Explorer Search Forum Online Help

**REST Request**

Endpoint https://gorest.co.in

Resource/Method: GET Users -> Users 1 [/public/v2/users]

Request

| Name          | Value | Style    | Level |
|---------------|-------|----------|-------|
| Authorization | QUERY | RESOURCE |       |
| Content-Type  | QUERY | RESOURCE |       |

Auth Headers (0) Attachments (0) Representations (0) JMS Headers JMS Property (0)

Raw [HTML] JSON XML

```

1 [
2   {
3     "id": 2587,
4     "name": "Datta Sethi",
5     "email": "datta_sethi@hilll.name",
6     "gender": "female",
7     "status": "inactive"
8   },
9   {
10    "id": 2585,
11    "name": "Gajaadhar Shah",
12    "email": "gajaadhar_shah@connelly.org",
13    "gender": "male",
14    "status": "active"
15  },
16  {
17    "id": 2584
18  }
19 ]

```

Headers (26) Attachments (0) SSL Info (3 certs) Representations (1) Schema (conflicts) JMS

Online Help

Contains - VALID  
Not Contains - VALID

Custom Properties REST TestRequest Properties

File Project Suite Case Step Tools Desktop Help

Empty SOAP REST Import Save All Forum Trial Preferences Proxy Endpoint Explorer Search Forum

**REST Request**

Endpoint https://gorest.co.in

Resource/Method: /Users -> Users 1 [/public/v2/users]

| Name          | Value | Style    | Level |
|---------------|-------|----------|-------|
| Authorization | QUERY | RESOURCE |       |
| Content-Type  | QUERY | RESOURCE |       |

Valid HTTP status codes Assertion

Specify codes

codes: 200

OK Cancel

Auth Headers (0) Attachments (0) Rep

26 Attachments (0) SSL Info (3 certs) Representations (1) Schema

Custom Properties

| REST TestRequest Properties |                      |
|-----------------------------|----------------------|
| Property                    | Value                |
| Name                        | REST Request         |
| Description                 |                      |
| Encoding                    | UTF-8                |
| Endpoint                    | https://gorest.co.in |

Contains - VALID  
Not Contains - VALID  
Valid HTTP Status Codes - FAILED  
-> Response status code:200 is not in acceptable list of status codes

Assertions (3) Request Log (1)

response time: 152ms (2268 bytes)

File Project Suite Case Step Tools Desktop Help

Empty SOAP REST Import Save All Forum Trial Preferences Proxy Endpoint Explorer Search Forum

**REST Request**

Endpoint https://gorest.co.in

Resource/Method: /Users -> Users 1 [/public/v2/users]

| Name          | Value | Style    | Level |
|---------------|-------|----------|-------|
| Authorization | QUERY | RESOURCE |       |
| Content-Type  | QUERY | RESOURCE |       |

Invalid HTTP status codes Assertion

Specify codes

codes: 500

Comma-separated not acceptable status codes

OK Cancel

Auth Headers (0) Attachments (0) Rep

26 Attachments (0) SSL Info (3 certs) Representations (1) Schema (conflicts) IM

Online Help

Custom Properties

| REST TestRequest Properties |              |
|-----------------------------|--------------|
| Property                    | Value        |
| Name                        | REST Request |
| Description                 |              |
| Encoding                    | UTF-8        |

Contains - VALID  
Not Contains - VALID  
Valid HTTP Status Codes - VALID  
Invalid HTTP Status Codes - VALID

Assertions (4) Request Log (1)

response time: 152ms (2268 bytes)

SapUi 5.7.0

File Project Suite Case Step Tools Desktop Help

Empty SOAP REST Import Save All Forum Trial Preferences Proxy Endpoint Explorer Search Forum

**REST Request**

Endpoint: https://gorest.co.in

Resource/Method: /Users -> Users 1 [/public/v2/users]

Request

| Name          | Value | Style    | Level |
|---------------|-------|----------|-------|
| Authorization | QUERY | RESOURCE |       |
| Content-Type  | QUERY | RESOURCE |       |

Configure Response SLA Assertion

Specify the maximum response time (ms)

200

OK Cancel

Auth Headers (0) Attachments (0) Representations (0) JMS Headers JMS Property (0)

Raw HTML JSON XML

```

1 [
2   {
3     "id": 2587,
4     "name": "Datta Sethi",
5     "email": "datta_sethi@hilll.name",
6     "gender": "female",
7     "status": "inactive"
8   },
9   {
10    "id": 2585,
11    "name": "Gajaadhar Shah",
12    "email": "gajaadhar_shah@connelly.org",
13    "gender": "male",
14    "status": "active"
15  },
16  {
17    "id": 2584
18  }
19 ]

```

Headers (26) Attachments (0) SSL Info (3 certs) Representation

Custom Properties

REST TestRequest Properties

| Property | Value        |
|----------|--------------|
| Name     | REST Request |

Assertions (5) Request Log (1)

## How to hit post/put

SapUi 5.7.0

File Project Suite Case Step Tools Desktop Help

Empty SOAP REST Import Save All Forum Trial Preferences Proxy Endpoint Explorer Search Forum

**step 1**

Projects

- REST Project 1
- REST Project 2
- reqres

**step2**

New REST Project

New REST Project

Creates a new REST Project in this workspace

URI: https://reqres.in/api/register

OK Cancel Import WADL...

SoapUI 5.7.0

File Project Suite Case Step Tools Desktop Help

Empty SOAP REST Import Save All Forum Trial Preferences Proxy Endpoint Explorer Search Forum

**Request 1**

Method: POST Endpoint: https://reqres.in Resource: /api/register

| Name | Value | Style | Level |
|------|-------|-------|-------|
|      |       |       |       |

Required:  Sets if parameter is required

Type:

Options:

Media Typ.. application/json   Post QueryString

```

{
  "email": "eve.holt@reqres.in",
  "password": "pistol"
}

```

step 1

step 2

step 3

Request Properties Request Params

| Property    | Value     |
|-------------|-----------|
| Name        | Request 1 |
| Description |           |
| Encoding    |           |

Auth Headers (0) Attachments (0) Representations (0) JMS Headers JMS Property (0)

Headers (0) Attachments (0) SSL Info Representations (0)

SoapUI 5.7.0

File Project Suite Case Step Tools Desktop Help

Empty SOAP REST Import Save All Forum Trial Preferences Proxy Endpoint Explorer Search Forum

**Request 1**

Method: POST Endpoint: https://reqres.in Resource: /api/register

| Name | Value | Style | Level |
|------|-------|-------|-------|
|      |       |       |       |

Required:  Sets if parameter is required

Type:

Options:

Media Typ.. application/json   Post QueryString

```

{
  "email": "eve.holt@reqres.in",
  "password": "pistol"
}

```

1: {
 2: "id": 4,
 3: "token": "QpwL5tke4Pnpja7X4"
}

Same for the PUT(just select put method)- we can apply assertions

here as well(post as well as put)