API Testing:

Client🡪InterNet-🡪Server

Client will send a url request to server then till we show the results;

There different types of Architecture:

1.single tier: example ms paint

2.2-Tier Architecture: Client---🡪Database example: games, music players

3.3-tier Architecture: client 🡪 Business logic /Application server 🡪database :all web Applications

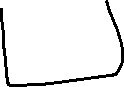
Client server Architecture:



Presentation layer(Scripting language used) 🡪Application layer(Programming language)) 🡪Data Layer(sql language)

🡪If we are testing in Presentation layer its called GUI Testing

🡪



* Without using UI/Presentation layer directly giving request from Business logic/Application layer is called API Testing
* API is collection of programs

Web Service?

If an API is on web then we called web service, after dev environment QA will test the api then it called API after that it will test for production we will put on web its called Web service

Difference B/T API&Web Services

* Web Service is an API wrapped in Http
* All web services are Api but not all api are webservices
* Web services need internet but api don’t need any network

Types of Web services:

2 types

🡪SOAP(Simple Object Access Protocol):only post xml operation will support

🡪RESTful web service(Representational State Transfer)/RestAPI Testing

Postman -1

GET(select)

POST(create)

PUT(Update)

DELECT(Delecte)

<https://request.in> --🡪Domain Name

//api/users? ------🡪Path parameters

Page=2--🡪query parameters

ALL together we call URL(uniform Resource Locater)

URI(uniform resource Identifier):complete path is called URI

Postman-2;

GET: URL is enough to retrieve the data

Post: request body/request payload

If we want to create table data use post

PUT: request payload/body

what ever we create data using post if we want to update that record we use PUT Command

DELETE: to delete

Every request will give a response as json format/html

tests["validating Status code"] **=** responseCode.code **==**200;

tests["Validating response body"]**=** responseBody.**has**("margin");

var response **=** JSON.**parse**(responseBody);

tests["page no"] **=**response.page **==**2;

for some json scripts are complex for that we need to download/add JSON Path extension to chrome browser

1.copy the actual path and paste in JSON PATH

2.click on find path the we get path

Example:

Var response =JSON.parse(responseBody);

Tests[“verify first name”] = response.data.first\_name ==”jamet”;

POST: using this we can create new data/record in database

Code to create data/record:

{

    "name" : "sindhu",

    "job"  : "shravan"

    }

Validation code for post :

var response **=**JSON.**parse**(responseBody);

tests["name"]**=**response.name **==**"sindhu";

tests["job"] **=**response.job**==**"shravan";

tests["code"]**=**responseCode.code**=**201;

to check the login details with post command:

<https://reqres.in/api/login>

in body🡪 Raw code

{

    "email": "eve.holt@reqres.in",

    "password": "pistol"

}

Validation code:

tests["token check"] **=** responseBody.**has**("token");

Create record:

<http://dummy.restapiexample.com/api/v1/create>

body 🡪 raw

{ "name": "Reddy", "package": "$200" , "age": "25" , "date": "01/01/2022"

}

Search the same record:

<http://dummy.restapiexample.com/api/v1/employee/20>

PUT Command: https://dummy.restapiexample.com/api/v1/update/20

{

        "id": 20,

        "employee\_name": "Dai Rios",

        "employee\_salary": "0000",

        "employee\_age": "25"

}

**Creating multiple set of data: Driven Testing:**

Verification points for header:

tests["status code"] **=** responseCode.code**==**200;

tests["check status in response"] **=** responseBody.**has**("success");

var response**=**JSON.**parse**(responseBody);

tests["check status is there"]**=**response.status**==**"success";

pm.**test**("check header", **function**()

{

    pm.response.to.be.header("Content-Type","application/json");

});

Note:

Body:

{

    "name" :"*{{name}}*",

    "Salary" : "*{{Salary}}*",

     "age"   : "*{{age}}*"

}

Note: attributes should same in file as well json code

We can use .csv/.json format files for data driven testing

Collection is called as suite and request is called as test case

***Command prompts execution***

1. Install node js Software

* Open Command prompt
* Type node -v
* Verify node js version/installation
* npm -v
* on top of the cmd we need to install newmen to generate the results
* command: npm install -g newmen

1. Install newmen

Npm install -g newmen

1. Export collection and then run from cmd prompt

Run Collection throw Command prompt

🡪1. newmen run filepath :🡪 to execute command and to get reports in json format

2. newman run filepath -r html 🡪 to get reports in html

We need to install npm html installer

Command: npm install newman-reporter-html

Medhod:3:using saharing url

3.newman run URL

4.newman run Url -r html (in case html reports)

**How to create own api**

Need to create json server on npm machine(command prompt)

Insatall json server

1.install json-server

Command: npm install -g json-server

2.create one info.json file

Command: json-server info.json

We will get urls in command prompt

**Postman Variables:**

**1.Collection variables:** collection variables used within the

Collection

* 1.Edit collection: variables 🡪 give variable name and URL

**2.🡪Edit collection 🡪 pre reqiset 🡪 give below command:**

postman.setGlobalVariable("url","http://localhost:3000")

* It will execute at and create at runtime

**2.Environment/Global variables:**

Go to manage environment variable settings: give variable name

It will allow to execute that particular environment

Sequence of request execution:

In pre request we need to hive code as

Postman.setNextRequest(“Request name”);

At the last request we need to give Test code bcz we don’t have any request

Code: postman.setNextRequest(“null”)

**How to do chain Request:**

API Request 1 🡪response 1 its used as input for Api2

API Reques1🡪Test 🡪

jsonData**=**JSON.**parse**(responseBody)

value**=**jsonData.data[0].first\_name

pm.globals.**set**("username", value)

-----------------------------------------------

API request2-🡪 Body 🡪

{

    "name": "*{{username}}*",

    "job": "zion resident"

}

**Authorization:**