NEGATIVE TESTING OF APIs

1. UNSUPPORTED HTTP METHODS : *Test the given API with an invalid/unsupported http method to validate if server returns 4xx response code [405]*
2. BROKEN ENDPOINTS TESTING : *Test the given API with an invalid endPoint to validate if server returns 4xx response code*
3. PAYLOAD TAMPERING TESTING : *Test the given API with an invalid payload such as invalid data type, data length, data field name etc to validate if server returns 4xx response code*

HOW DOES MY FINAL PROJECT STRUCTURE LOOKS LIKE ?

PERFORMANCE TESTING OF APIs

* Postman tool can be used to test the APIs performance upto a limit of vUsers=100 & load time of 60min
* We capture the response times
* We also capture the response code & ensure it is always 2xx
* v1 user : create user > get > update > get > delete > get
* v2 user : create user > get > update > get > delete > get
* v3 user : create user > get > update > get > delete > get
* v4 user : create user > get > update > get > delete > get
* v5 user : create user > get > update > get > delete > get

CONTINOUS TESTING OF APIs

*It is the process of executing the Postman test scripts at a pre-defined scheduled day & time*

SHARE COLLECTIONS TO ANOTHER WORKSPACE

* Move

EXPORT/IMPORT COLLECTIONS

* Export
* Import

USAGE OF VARIABLES IN POSTMAN TOOL

* A variable acts as a place-holder to store the value of an object
* 3 types of variables in Postman tool
  + Collection variables : *scope is to the collection itself*
  + Global variables : *scope is across all the collections in the same workspace*
  + Environment variables : *scope is across all the collections in the same workspace and the context here is environment specific*
* To create variables in Postman tool,
  + Step by step wizard
  + Manual process
  + Using java-scripts
    - Set > to create a variable
    - Get > to read the variable value
    - Clear > to delete the variable
* DEV-ENV
  + Protocol = https
  + baseURL = gorest.co.in
  + Name=dev-user
  + Email=devuser@gmail.com
  + User-Agent=Windows + Edge
* QA-ENV
  + Protocol = https
  + baseURL = gorest.co.in
  + Name=qa-user
  + Email=qauser@gmail.com
  + User-Agent=Linux + Firefox
* UAT-ENV
  + Protocol = https
  + baseURL = gorest.co.in
  + Name=uat-user
  + Email=uatuser@gmail.com
  + User-Agent=Mac + Safari

INTEGRATION OF POSTMAN TOOL WITH POSTBOT-AI FEATURE

* Generate Tests
* Postbot prompt

INTEGRATION OF POSTMAN TOOL WITH NEWMAN TOOL [CLI mode of testing]

* Newman is a CLI interface plugin developed by Postman
* Newman is used to generate test reports in various formats such as *cli, json, xml, html*
* Newman CLI can be used to integrate Postman API automation test scripts with CI/CD tools such as Jenkins
* Newman pre-req:
  + NodeJS
    - <https://nodejs.org/en/download>
    - node --version
  + Newman
    - npm install –g newman
    - newman --version
  + HTML test report capability
    - npm install -g newman-reporter-htmlextra
* To generate CLI test reports, the command is

newman run postmanCollectionPath –r cli

* To generate JUNIT test reports, the command is

newman run postmanCollectionPath –r junit

* To generate JSON test reports, the command is

newman run postmanCollectionPath –r json

* To generate HTML test reports, the command is

newman run postmanCollectionPath –r htmlextra

1. Download PostmanCollection (export) to your PC
2. Open command prompt at above path
3. Type the required newman command to run Postman collection
4. Analyze the test report generated

Newman additional commands

To generate a custom title in the generated report, the command is

newman run postmanCollectionPath –r htmlextra --reporter-htmlextra-title "Project Name"

To run a specific folder in the collection, the command is

newman run postmanCollectionPath --folder “folderName” –r htmlextra

newman run postmanCollectionPath --folder “folderName” --folder “folderName” --folder “folderName” –r htmlextra

To run a specific folder multiple times in the collection, the command is

newman run postmanCollectionPath --folder “folderName” –n “iterationsCount” –r htmlextra

To run data driven automation testing in the collection, the command is

newman run postmanCollectionPath --folder “folderName” –d “pathOf.csvfile” –r htmlextra

To run environment variables specific folders in the collection, the command is

newman run postmanCollectionPath --folder “folderName” –e “pathOfEnvFile” –r htmlextra

INTEGRATION OF POSTMAN TOOL WITH GITHUB

* Github is from Microsoft
* It is used for version control mgmt of software
* It is used to take backup of the automation code
* Sign up to github.com > sign in to github.com
* Create a github repo, where we can backup our automation code

Approach#1- using github.com without git

* Click on ‘upload existing files’ > select ‘collection.json’ file > click on upload
* Now, the postman automation code is backed up on github.com repo

Approach#2 – using git & then pushing to github.com

* git config --global user.name "Kiran"
* git config --global user.email "kiran@gmail.com"
* git init // initializing git repo
* git status // asking for the status
* git add . // track all the untracked files/folders
* git commit –m “v1 automation code” // to commit all tracked files/folders
* git branch -M main // switch to the branch ‘main’
* git remote add origin <https://github.com/kirangopisetty/7AM-Training.git>
* git push -u origin main

Approach#3 – using Postman & github direct integration

* On the Postman tool, click on Home > click on Integrations > click on ‘Browse all integrations’ > click on Github > click on ‘Add integration’ button for ‘Backup a collection’ >

INTEGRATION OF POSTMAN TOOL WITH JENKINS

* Jenkins is a server-side tool used by Dev team, QA team, DevOps team
* Jenkins is built using Java [pre-req: Java v11, v17, v21]
* Jenkins is an open-source/free software
* Download Jenkins.war from <https://www.jenkins.io/download/>
* Jenkins is used by Dev team for CI of source code
* Jenkins is used by QA team for CT of new code changes deployed
* Jenkins is used by DevOps team for CD of the new tested code in the next environment
* Jenkins workflow: CI > CD > CT > CD
* To run Jenkins, execute the below command from the path where jenkins.war is downloaded
  + java –jar jenkins.war
* UN: admin
* PWD: ea0e581cbac64514a0099afa8356bb81
* Jenkins URL: <http://localhost>:8080

PROJECT#2 – SWAGGER BASED REST APIs

* API documentation format: Swagger
* Swagger documentation URL: <https://httpbin.org/>
* Module: Auth
* # of APIs developed: 3
* Algorithms used for authentication:
  + Basic-Auth: *This auth expects the user to enter valid UN & valid PWD along with API request so that we can receive a valid server response*
  + Bearer token Auth: *This auth expects the user to enter valid bearer access token along with API request so that we can receive a valid server response*
  + Digest Auth: *This auth expects the user to enter valid UN, valid PWD & valid QOP [quality of protection] along with API request so that we can receive a valid server response*
* Exp response: 200 OK ; json body ; application/json
* Basic Auth API: UN: user ; PWD: passwd
* Bearer Access token Auth API: Token: Bearer 12345
* Digest Auth API: UN: user ; PWD: passwd ; QOP: auth or auth-int

PROJECT#3 – SOAP APIs

* Simple object access protocol
* HTTP methods allowed = POST
* HTTP methods not allowed = GET, PUT, PATCH, DELETE etc
* Request body/payload = xml format
* Response body = xml format
* API documentation = wsdl file or xml
* WSDL = web services description language
* API documentation URL: <http://dneonline.com/calculator.asmx?wsdl>

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

* 4 APIs are developed
  + ADD
  + SUBTRACT
  + MULTIPLY
  + DIVIDE

IMPORT WIZARD

Using import feature in Postman tool, we can convert the API documentation into Postman collections

* WSDL
  + <http://dneonline.com/calculator.asmx?wsdl>
* Swagger-json
  + <https://demo.testfire.net/swagger/index.html>
  + <https://petstore.swagger.io/>
* Swagger-yaml
  + <https://editor.swagger.io/>
* cURL
  + <https://blazedemo.com/login>
  + <https://blazedemo.com/password/reset>
  + <https://blazedemo.com/register>
* Github repo
* HAR
  + http archive resource
  + <https://blazedemo.com/login>
  + <https://blazedemo.com/password/reset>
  + <https://blazedemo.com/register>

POSTMAN INTERCEPTOR

It is a chrome extension on the browser which can intercept your web application calls & captures the APIs

<https://katalon-demo-cura.herokuapp.com/profile.php#login>

* Login
* Book an appointment
* Logout

Conversion from API format to cURL format

Conversion from cURL format to API format

POSTMAN SETTINGS, TROUBLESHOOTING & LEARNING CENTER

* Understanding the Postman History feature & its usage [Save Responses, Delete, Clear All, Add]
* Learn how to debug & troubleshoot API testing using Postman tool & any browser
* Introduction to Postman console
* Logging feature to debug automation scripts
  + console.log | console.debug | console.info | console.warn | console.error
  + Learn how to copy, search or clear all the logs from Postman Console
* Exploring the recorded logs using *View postman logs in explorer*
* Discovering Postman configuration & all execution logs at **\AppData\Roaming\Postman**
* Other features: *Notifications, Manage Notifications, Find & Replace, Boot Camp, Runner, Trash, Two Pane view, Postman Community & Learning Center, Global search, WorkSpaces sync with Postman Servers*
* Explore Postman Settings [*General, Themes, Shortcuts, Data, Add-ons, Certificates, Proxy, Update, About*]
* Tracking Postman’s status via [https://status.postman.com/#](https://status.postman.com/)