6. How to approach API automation test and scope. Which tools are you using for automation testing?

Answer to the Question Number -6

API Testing Approach:

Testing for APIs is similar to testing any other software, except that you need to think a bit more technically. Unlike GUI based systems, there are no text fields or buttons which you can test, you have to essentially hit the backend calls that are made when a user would perform an action. Since APIs help systems to integrate and communicate, it is important that you check each system endpoint where an API call can make impact.

In order to effectively test an API, the tester should first determine the functional scope of the API. Next the test environment should be set up with proper configuration of servers, databases and every resource the API interacts with.

The test cases designed to test an API should focus around the points given below:

1. **Input validation -** Test the API with different input parameters and verify the response. The data, response code, message etc should be correct for each set of input parameters. Does the API return correct HTTP error codes like 200 for valid and 400 for invalid input parameters?

2. **JSON format validation -** Verify that the JSON or XML response of the API is correctly structured.

3. **Business Logic -** Is the API doing the task it is supposed to do? Suppose we have a fetch Balance API that gives the current balance for a user. If it

returns the correct HTTP response code, but the actual balance being returned for user is of previous month, the business logic of this API is broken.

4. **Negative test cases -** Hit the API with incorrect/invalid parameters, missing/extra values, null values for mandatory fields and observe the output. Is the API able to gracefully handle unreasonably large amount of payload data, special and non-ASCII characters, long strings and integers, incorrect data types for parameters? Are there any buffer overflow conditions? How does it behave in case of conditions like timeouts, server failures, etc.? Is the exception handling mechanism in place? Are the error messages clear and relevant?

5. **Reliability tests -** Check whether the API can consistently return correct response, or do response failures occur often.

6. **Call sequencing checks -** If the output of the API being tested includes modification of a data structure, change of a resource state (Like a DB record), firing of an event or call to other APIs, this should be functioning correctly.

7. **Security testing -** Every business critical API should undergo security testing techniques, to ensure that its code and associated features cannot be accessed and used by unauthorized users.

8. **Performance testing -** Verify if the response time of the API calls is too high or varies considerably. Also we can verify if the performance of the API response degrades when called using a large number of clients.

API Testing Scope:

A good approach will be to increase the scope as the testing progress successfully. Organize the API testing at each level will help in achieving this. Testing the contract can be an example of level one testing. Such tests should validate that the agreements are documented as per the specifications and can consume those API. Successful completion of this should be the gateway for the next level of testing.

Each API request and response must be validated individually; these cases should form the minimal building blocks for API testing. For example, start with POST request to create a resource. Upon successful creation of the resource, the response will include an auto-generated id. Just like the previous level, the successful completion of these cases should open the quality gate for the next level of API testing.

During level three focus should be to test a series of requests mimicking user actions. For example, start with POST a a request to create a resource. Upon successful creation of the resource, the response will include an auto-generated id. In the following request, use this previously generated id and fire a GET request to retrieve the details of the newly created resource. Followed by request to update the details of a newly created resource. As a next step, you can remove that resource using the DELETE request. Finally, to ensure the resource no longer exist by calling a GET request.

Can take the API testing even to the next level where the API testing integrates with automated UI testing. Incorporating both types of testing can leverage the benefit of API testing and UI testing, and both the testing can bring down pitfalls. For example, API testing is quick, and UI testing can be time-consuming, so integrating them can increase the coverage and reduce the timing if performed individually.

I am using tools Postman, Rest Assured and Jmeter for Api testing and also using Selenium for automation testing and also using Selenium Framework Selenium web driver, Junit and TestNG.

Answer to the Question no- 7

TPS (Transaction Per Second):

TPS (Transaction Per Second) = Number of users/Ramp-up period (seconds)

= 3200/600

= 5.3333