What to do for successful API Testing

- **Categorize the test cases**

An application may involve several APIs which may sometimes run into triple digits. In order to maintain the test cases for quick reference and execution, it is a good idea to categorize them.

- **Prioritize the API calls**

Calls made to API's should be prioritized for ease of testing.

- **Automate the API testing process**

API tests are usually deemed stable and major changes are done mainly when business logic is changed. This makes them perfect candidates for test automation. Since API testing is a type of Black-box testing different combinations of inputs can be used to test a scenario, making this a Data driven test approach. The test data, scripts and API endpoints can be saved for execution at a later stage.

- **Choose a suitable Automation Testing Tool**

There are several tools, that are available in the market for API testing. Choose a suitable tool carefully to leverage the benefits of automation.

- **Create positive and negative tests**

For a complete scenario check API testing requires to run both, positive and negative tests. Since API testing is a data driven approach, various combinations of data inputs can be used to test these. In either case, proper care must be taken to ensure that API returns appropriate response to the calling function.

- **Share the test results**

API testing failure should be brought to the attention of all the stakeholders involved on a priority basis. Even better approach would be to send out immediate notifications using team notification channels in event of any failure. This will help the team in taking prompt action.

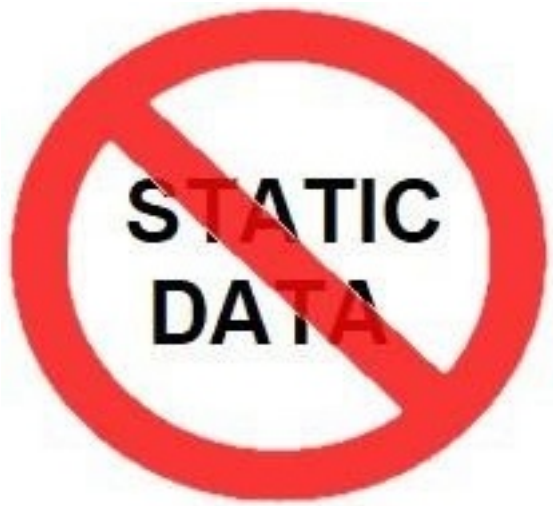
- **Perform Load Tests**

An API should handle load tests either by functioning as expected or failing gracefully in a predictable manner. It should be able to handle a variety of input data and manage error conditions, in case of failure.

What to avoid during API Testing

Following points should be taken care of while performing API testing.

- **Using static data**



In a real life scenario, API endpoints rely on inputs from calling code or on the outputs of other API's. Using static data embedded in code or otherwise will yield incorrect results when the API is called in the live environment. It is prudent to perform a proper Integration testing of API's by following actual user and data flows.

- **Ignore dependencies**

API's often depend on other API's and sometimes on external services also. It is vital to test the third party API's in test environment and then test the dependent API along with those API's to have a holistic picture. In a nutshell,

the whole ecosystem needs to be tested in order to ensure that any changes/ upgrades in other API's does not impact the functioning of dependent API's.

- **Missing the response time assertions**

It is important to keep track of the response time of API calls. While a test may pass functionally, there could be a number of reasons for delayed response. This affects the end user experience. Any application which takes too long to compute or load or give results is not what the end user wishes to use. Hence, it is a good practice to include a time based test to check the response time. If API is taking too long to respond then it should be reported so that the developers can look into the code to understand the delayed response.

- **Treating security testing lightly**

API's like any other code are vulnerable to threats and attacks. It becomes even more important to conduct security tests when a third party free/paid API is used. Hence, it is important to test for all possible loopholes and seal them.

- **Ignore One Time function Calls**

Extra care must be taken while making one time function calls like Delete, CloseWindow etc. These have direct effect on the application and must not be treated lightly.

- **Keep documentation low on priority**

It is good practice to document all the test cases in detail so that they become a ready reckon-er guide and can be used by anyone in the team.