AT*SQA

API Testing: Environments, Tools, and Future Proofing Micro-Credential

Syllabus

AT*SQA
MICRO-CREDENTIAL
API Testing
Environments, Tools, and Future Proofing

Table of Contents

General Information

KEYWORDS

cURL, environment, Powershell, Postman, ReadyAPI, rapid growth, change management

LEARNING OBJECTIVES FOR API TESTING: ENVIRONMENTS, TOOLS, AND FUTURE PROOFING

API Test Tools
Explain examples of off the shelf and open source tools for API testing
Explain examples of in-build API test tools
Summarise gRPC and graphQL test tools

Test Environments
Summarize the considerations regarding tool selection for performance testing APIs

Performance Test Tools and Support
Summarize the considerations regarding tool selection for performance testing APIs
Explain the advantages and disadvantages of cloud-based performance tools
Summarize the usage of jMeter in API testing

Test Automation
Explain why test automation is used in API testing
Explain the environment considerations in API test automation
Summarize the skills needed for the API tester when developing test automation

Expect Rapid Growth
Explain how the rapid development cycles affect testing

Build for Change

Summarize the considerations for building a test approach for change
Explain why maintainability of tests is necessary

Plan for the Future

Explain how API testing can be made more efficient

Anticipating the Future

Explain the factors for a tester to consider when planning for the future of API testing

Environments and Tools

Tools

There are both off-the-shelf licensed API test tools as well as a wealth of open source or freeware API test tools. The tester will need to understand the license requirements of the organization, licensing budgetary restrictions, licensing requirements (e.g., node locked), and the in-team comfort/training with a particular tool.

Most API tools are lightweight requiring small amounts of hard drive space and RAM but do require local machine installation. While some tools are only installable on Microsoft Windows machines, many others can be installed on Microsoft Windows, iOS, or Linux.

It's advisable to conduct a proof-of-concept to evaluate a selected set of tools to ensure they will work in the test environment and with the software under test. Building a pro vs con list of the different tools is helpful to select the best tool and to justify any licensing costs. While evaluating the tool itself, also evaluate the vendor to understand what support they will offer, available documentation, reliability, etc.

OFF THE SHELF LICENSED TOOLS

Smartbear's ReadyAPI is an off the shelf licensed API testing tool that performs API testing as well as performance testing. The tool is installable on Microsoft Windows®, iOS, or Linux. Running tests on a remote machine (i.e., a VM in the cloud) requires a 'runner' TestExecute be running on the remote machine. The tool does have Azure DevOps integration.

OPEN SOURCE OR FREEWARE TEST TOOLS

SoapUI is the open source version of Smartbear's ReadyAPI. SoapUI has a performance testing capability that allows the tester to set up multiple virtual users easily and with no cost. It allows significant configuration capabilities to start the transactions and to gather the timing metrics. It does not, however, have a capability to read data from a data file which limits the types of tests that can be run.

SoapUI is not as widely used as Postman. Postman has emerged in the industry as the 'go-to' tool for API testing. Postman has some test integration with Azure DevOps but requires some set-up. (Jan Kratochvil, 2023)

IN-BUILD OPERATING SYSTEM TEST TOOLS

Operating systems have 'in-build' (i.e., tools that install with the OS install), tools that have the ability to test APIs. This approach is not often used as using the tool requires specific tool knowledge and scripting skills.

PowerShell can be used to test REST APIs or graphQL queries on Windows. However, there is not PowerShell support for gRPC. (James Newton-King, 2022)

cURL (Client URL) is a well-known tool that can be used to test APIs. cURL is installed by default in iOS and Linux. cURL can be installed in Windows. It works through a command line interface and can be used to send API requests. Requests normally consist of the endpoint, the HTTP method (GET, POST, PUT, DELETE), the header and the body.

GRPC TEST TOOLS

Like REST API tools, there are freeware and licensed tools that can be used to test gRPC services.

ReadyAPI supports gRPC testing. There are also several freeware alternatives to licensed gRPC test tools. The most popular freeware tool to test gRPC is Postman. In addition to Postman, gRPCurl is an open-source command-line tool that provides interaction with gRPC services. gRPCui can be used to build on top of gRPCurl and adds an open-source interactive web UI for gRPC.

(Newton-King, 2022)

GRAPHQL TEST TOOLS

There are several graphQL tools available including the following:

- GraphiQL: GraphiQL is an in-browser IDE for exploring GraphQL APIs.
- GraphQL Voyager: GraphQL Voyager is a tool that allows the tester to represent any GraphQL API as an interactive graph.
- Hygraph: Hygraph is a tool that allows the tester to build a GraphQL Content API in minutes.
- GraphQL Faker: GraphQL Faker is a tool that allows you to mock or extend your GraphQL API with faked data. (Doerrfeld, 2017)

Environments

ENVIRONMENT CONSIDERATIONS

When testing API(s) the tester will need to consider the environments. Different API versions can be deployed to different environments. Conversely, a version of an API can be deployed to all environments simultaneously. Each build and deployment approach has its own test considerations. The tester will need to have clear communication from stakeholders (such as the release manager) to keep testing in sync with the build deployments.

Data Handling

Testing for APIs must consider receiving data and sending data. Data must be handled with the appropriate level of security and reliability. The tester may need to work with other stakeholders such as data analysts or data scientists to ensure the test data is representative of real production data – without the need for using sensitive or personal information. Data may have an underlying tight coupling with the environment under test. The tester may need to consider data clean-up or resetting the data so other tests or processes are not disrupted.

Performance Test Tools and Support

OFF-THE-SHELF LICENSED PERFORMANCE TEST TOOLS

There are off-the-shelf licensed API test tools that include some performance test functionality. There are also licensed tools that focus on performance testing that cater to API testing. Additionally, there are freeware tools that can be used for performance testing.

When selecting an API testing tool, its ability to perform some degree of performance testing needs to be assessed. Before selecting a performance test tool, the tester will need to understand the license requirements of the organization, licensing budgetary restrictions, licensing constraints (such as a finite number of virtual users), and the team's comfort/training with tool use.

The tester will also need to be aware of restrictions for each machine that may generate traffic used in the performance testing. These restrictions might include the OS supported for tool installation, the license requirements for each machine, and the monetary cost of machine use (such as a VM in the cloud). For example, if the performance scenario requires 5 different (virtual) machines generating traffic for an API, each of the 5 machines may need a license and may incur cost while the VM is running.

CLOUD-BASED PERFORMANCE TESTING TOOLS

There are performance test frameworks available for cloud service providers such as BlazeMeter® or Azure®. The cloud providers offer a means of traffic generation and report generation without the overhead of VM or container management. Cloud-Based providers do not tend to offer good ROI (return on investment) for small performance test projects. For large performance testing requirements, the tester will need to keep these service options in mind.

OPEN SOURCE AND FREEWARE TEST TOOLS

Though there are open source and freeware tools, the industry ‘go-to’ tool is jMeter™. jMeter has an extensive user base and instruction base (using YouTube for example). jMeter can be used for simple and complex performance test scenarios. It is suggested that if a tester has not used a performance test tool before, that the tester study and use jMeter in conjunction with other learning materials to build a performance test understanding.

Test Automation

API automated testing is strongly encouraged as the nature of APIs lends itself to the use of automated testing code to test the API code. API tests are often repeatable and tend to require less overhead to add and maintain tests (versus WebUI tests, for example). Tools such as Postman feature the ability to generate code from API calls (called a collection in Postman). AI tools such as ChatGPT can generate API test code scaffolding in the preferred language in seconds.

TEST AUTOMATION TOOL SUPPORT

The test automation expectation is that tests can be automated and executed in a repeatable, durable, reliable, and reportable fashion. The automated testing tools will need to accommodate execution via (test) build pipelines that might be run several times a day. A proof-of-concept should be used to determine the automation’s fit for use.

Test Automation Test Environment(s) Considerations

When designing a test automation framework, there are several considerations that should be weighed:

- The test automation may be required to execute in multiple test environments in parallel.
- Test data dependencies (i.e. different data needed in different environments) is a consideration when designing and developing the test automation. Test configuration (via a configuration file consumed by the test) may be a method of handling different environments
- If there are multiple test automation execution environments, segregation or aggregation of the results may be helpful to determine in which environment(s) a failure occurred. For example, if a test automation run were to occur in environment A and B, a separate report, or a way to differentiate the test results, will be required to understand if test failures occurred in A, B, or both A and B.

SKILLS NEEDED

As with any test automation project, programming and scripting skills are needed to develop high quality, maintainable test scripts. API test scripts might be leveraged to be used for API performance testing; hence, any previous performance test experience will be an asset when developing the test scripts. A tester will need to understand the tool or code enough to implement configurations that result in both futureproofing as well as allowing the scripts to run in different environments. AI has become a way to expedite test script creation; however, the tester will need the skill to know when and where an AI gives incorrect scripting code so that the scripts can be fixed for use. If the tester is new to API test script development, it is advised that a peer review is conducted on the scripts so the tester's skills can be refined.

Future Proofing

Expect Rapid Growth

APIs can be developed more and more quickly given the current code scaffolding and use of AI. The tester will have the expectation to develop API tests in sync with development so as not to be the bottleneck for product and feature releases. Therefore, the tester will need to understand, plan for, and adapt to the rapid test expectations placed on them.

Build for Change

As profit margins decrease due to competition in the market, pressure will be put on the development and testing teams to produce high quality, maintainable products quickly. Testing will need to engage early with development to plan the testing, procure the tools and environments, and upskill as needed.

ARCHITECT THE TESTING

It will not be enough for the developers to plan for change. The test approach must also be architected for change. Some of the considerations for this flexible framework include:

- Implementing or utilizing test environments that can be quickly assembled and disassembled
- Utilizing the most appropriate tool for the task by using different tools for different tasks if it is pragmatic to do so
- Where possible, conform to the tool and code language governance of the product team
- Implementing maintainable, repeatable test automation steering clear of data dependencies where possible
- Designing the performance testing approach at the beginning of a project
- Conducting performance testing throughout the software development lifecycle
- Conducting security testing throughout the software development lifecycle
- Building a robust test result reporting structure for auditability and coverage metrics
- Accommodating a risk-based testing approach
- Monitoring the ROI

Tools must be selected for flexibility and the ability to adapt to the changing market. Tool vendors must be highly responsive to the market changes. Relying on past reputation will not be sufficient in a market that is moving quickly and has many tool options.

Re-usability is a goal. As APIs tend to underpin a product's back-end, initial investment in re-usable test code provides a better ROI.

ENABLE EFFICIENT MAINTENANCE

Efficient maintainability is a requirement. The test environments, tools, and test scripts must be able to be maintained or replaced. Test tools and scripts might be developed in the short term, but the scripts might be used well into the future. Discussions with the product/project manager on the projected changes are advised to make an informed decision on how to build the tool set or test scripts for efficient maintenance.

SELECT TOOLS FOR FLEXIBILITY

Tools will need to be selected to attempt future proofing and flexibility. As the API endpoints evolve, the test tools will need to accommodate the additional testing requirements while keeping the current test coverage in-place. The tool may need to provide a pathway to automated testing. For example, Postman offers the ability to create test code from an existing collection in a selection of languages. Tool characteristics such as licensing costs and support availability must be monitored to ensure efficient long-term use.

Plan for the Future

Agile and iterative lifecycles already dominate the industry, but new leaner methodologies may become popular. The tester must remember that any lifecycle model will require some adaptation and the tester must understand the moment of involvement and level of involvement that will be expected in the various models.

Timescales for the development and testing of applications have never been so short. The market demand has never been so high. The tester must expect to work with the developers to ensure the best possible product is being released within the given timeframe. In addition, any initial investment of time and effort to make future test development more efficient is advised. Using configuration settings or parameters where possible might take some time to implement initially but will pay rewards during the life of the product in terms of reduced maintenance.

While new lifecycle models may be on the horizon, there will still be a need for efficient testing. This means taking a lightweight approach to deliver a proper ROI. Automated functional testing, security testing, and performance testing may be required, and the tools must be appropriate to the needs, schedules, and budgets. Tools will likely adapt to fill the market as well.

Anticipating the Future

Though REST, gRPC, and graphQL have been around for over or nearly a decade (graphQL was released in 2015), testers need to be ready and willing to adopt new technologies, investigate new tools and learn more efficient and leaner testing methodologies. Testers cannot anticipate the effect of new technologies such as AI, but testers can study and adapt to the ever-changing software testing landscape.

References

Acknowledgements

This document was produced by a core team from the AT*SQA Foundation Level Working Group – API Micro-Credentials

Judy McKay (chair).
Johnathan Seal – Author
Earl Burba, Judy McKay, Randy Rice - Reviewers

The core team thanks the review team for their suggestions and input.

AI (such as ChatGPT) was not used to create content for this document.

Purpose of this Document

This syllabus forms the basis for the Association for Software Testing and Quality Assurance (AT*SQA) Micro-Credential for API Testing. This particular syllabus is focused on the Introduction and the Test Planning and Design areas.

AT*SQA provides this syllabus as follows:

1. To training providers, to produce courseware and determine appropriate teaching methods.
2. To credential candidates, to prepare for the exam (as part of a training course or independently).
3. To the international software and systems engineering community, to advance the profession of software and systems testing, and as a basis for books and articles.

The AT*SQA may allow other entities to use this syllabus for other purposes, provided they seek and obtain prior written permission.

Trademarks

Works Cited

James Newton-King. (2022, 09 22). Compare gRPC services with HTTP APIs. Retrieved from https://learn.microsoft.com: https://learn.microsoft.com/en-us/aspnet/core/grpc/comparison?view=aspnetcore-8.0

James Newton-King. (2022, 09 20). Test gRPC services with Postman or gRPCurl in ASP.NET Core. Retrieved from learn.microsoft.com: https://learn.microsoft.com/en-us/aspnet/core/grpc/test-tools?view=aspnetcore-7.0

Ana. (2023, 1 19). What are API Headers. Retrieved from mixedanalytics.com: https://mixedanalytics.com/knowledge-base/api-headers-explained/#:~:text=in%20API%20Connector%3F-,What%20are%20API%20headers%3F,an%20API%20key%20for%20authentication.

contributors, M. (2022, 9 9). Accept. Retrieved from developer.mozilla.org: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Accept

Esteban Herrera. (2022, 08 24). GraphQL vs. REST APIs: Why you shouldn't use GraphQL. Retrieved from https://blog.logrocket.com/: https://blog.logrocket.com/graphql-vs-rest-api-why-you-shouldnt-use-graphql/

Hitesh Baldaniya . (2021, 07 19). Why and When to Use GraphQL. Retrieved from https://dzone.com: https://dzone.com/articles/why-and-when-to-use-graphql-1

Jan Kratochvil. (2023, 01 13). A new Postman integration for Azure DevOps users. Retrieved from https://blog.postman.com/: https://blog.postman.com/postman-integration-for-azure-devops-users/

jwt.io. (n.d.). introduction. Retrieved from jwt.io: https://jwt.io/introduction

Levy, T. (n.d.). A Machine Learning Approach to Log Analytics. Retrieved from logz.io: https://logz.io/blog/machine-learning-log-analytics/

MDN contributors. (2022, 10 23). Content-Type. Retrieved from developer.mozilla.org/: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Type

Microsoft. (n.d.). Data contract overview. Retrieved from https://learn.microsoft.com: https://learn.microsoft.com/en-us/industry/retail/intelligent-recommendations/data-contract

MIcrosoft. (n.d.). JavaScriptSerializer.MaxJsonLength Property. Retrieved from learn.microsoft.com: https://learn.microsoft.com/en-us/dotnet/api/system.web.script.serialization.javascriptserializer.maxjsonlength?view=netframework-4.8.1

Mozilla. (n.d.). HTTP response status codes. Retrieved from mozilla.org: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status mozilla.org. (n.d.). Synchronous and asynchronous requests. Retrieved from mozilla.org: https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/Synchronous_and_Asynchronous_Requests

RedHat. (2020, 5 8). What is a REST API? Retrieved from www.redhat.com: https://www.redhat.com/en/topics/api/what-is-a-rest-api

Redhat.com. (2022, 6 2). What is an API? Retrieved from www.redhat.com: https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces

RFC. (n.d.). RFC-2616. Retrieved from RFC-2616: https://www.rfc-editor.org/rfc/rfc2616#section-6

Tamura, K. (2015, 04 21). The log: The lifeblood of your data pipeline. Retrieved from www.oreilly.com: https://www.oreilly.com/content/the-log-the-lifeblood-of-your-data-pipeline/

Tesauro, M. (n.d.). API Security Tools. Retrieved from owasp.org: https://owasp.org/www-community/api_security_tools

W3 Schools. (n.d.). JSON Data Types. Retrieved from W3 Schools: https://www.w3schools.com/js/js_json_datatypes.asp

Tesauro, M. (n.d.). API Security Tools. Retrieved from owasp.org: https://owasp.org/www-community/api_security_tools

W3 Schools. (n.d.). JSON Data Types. Retrieved from W3 Schools: https://www.w3schools.com/js/js_json_datatypes.asp

AT*SQA

MICRO-CREDENTIAL

API Testing

Environments, Tools, and Future Proofing

www.atsqa.org