**Codeless Test Strategy**

Written By: Edward Davies

Self-Published on: June 2022

Summary

The primary objective will be to analyse new test automation and provide a clear picture for the benefits of codeless test automation to achieve high quality and enhanced test accessibility.

We will evaluate our current test strategy and then we will explain the methods of how information were gathered.

We will explore the key findings in detail and compare different test automation tools available on the market.

In closing, we will conclude by providing 4 options for the new test automation.

# Introduction

Currently we perform manual testing on multiple environments; (i.) manual testing on a smoke environment, (ii.) regression testing on a formally deployed test environment, and (iii.) user acceptance testing on a beta environment. Manual regression testing saps time and is susceptible to human error so it is ripe for automation. In contrast, Application Programming Interface (API) tests are automated on our smoke environment, but are limited to developers who have the skills required to maintain the tests. Currently, we have two API test projects, the first uses Java REST Assured and the second uses a JSON parser (Martin, 2022; Redwood Logistics, 2021).

End-to-End (E2E) test automation is performed on the smoke environment using Google Protractor and TypeScript. However, Protractor won’t be updated after the end of 2022 because Google decided there are other robust E2E options now available, so it did not need to continue maintaining its own test tool. There is another E2E test project, which was migrated to Microsoft Playwright. Both API and E2E test automation require experienced developers to design, maintain, and run the tests (McDaniel, 2022; Playwright, 2022).

Tests should be automated and run continuously on a Continuous Integration (CI) pipeline in order to find problems. We already implemented API and E2E testing on the CI pipeline for the core software. However, for test automation to succeed, everyone must create high quality work for a consistent outcome. Therefore, the objective should be to realise the power of test automation, introduce Artificial Intelligence (AI) self-healing tests, become cloud-ready, enhance test coverage, ease our test maintenance, and provide real-time reporting (Bigelow, 2021).

# Methodology

Research for this paper was primarily derived from new test automation evaluation, team insight, proof of concepts (POCs), team discussion, whitepapers, and best practice (Duffy, 2022; Yeh *et al.*, 2009).

# Key Findings

## Key Finding #1: Cucumber is not required for automation testing.

Writing Cucumber scenarios is widely considered the standard for defining software behaviour but developer skills are still needed to translate those scenarios into actual tests (Unadkat, 2021).

Behaviour Driven Development (BDD) and Cucumber were designed to eliminate misunderstanding of how software is meant to work. Cucumber also addresses gaps in team communication and ensures correct understanding of requirements (Cucumber.IO, 2016; Pryce, 2020).

However, for automation Cucumber adds complexity and extra dependency that can potentially break during testing and in some cases increased time to test. Furthermore, Cucumber’s own founder, Gáspár Nagy, suggested there is minimal yield to use Cucumber for test automation, because it results in brittle tests and costly test maintenance (Advolodkin, 2020; Rose, 2021; Zhan, 2021).

## Key Finding #2: Codeless testing for fast-moving projects.

According to research and forecasting 2021-2031 at Future Market Insights (FMI, 2021), codeless testing is preferred on agile projects. The latest “State of Testing” report in March 2022 surveyed 900 testers and concluded that demands for test automation are increasing on agile projects, while time for testing decreased (PractiTest, 2022).

The company, Applause, recently surveyed 2000 product owners, software developers, as well as Quality Assurance (QA) and DevOps professionals and found that lack of experience was still the main challenge in test automation (Applause, 2021).

Given the rapidity of project deliveries and shrinking test time, we should consider codeless platforms. Codeless automation can enable us to focus on testing, reduce learning curves, and make test automation widely accessible (A1qa, 2022; Testsigma, 2022).

The major downsides of codeless automation include limited test re-usability and scope, inability to debug, advanced testing still requiring coding ability, vendor lock-in, and high licence costs (Knoldus, 2022; Rana, 2020).

Furthermore, codeless isn’t strictly zero code, scripting is still happening behind the scenes for tests. However, newer codeless automation has tried to address these limitations by improving test re-usability, having structured organisation of tests, and allowing testers with developer skills to code tests if required (Bqurious, 2021; Nizri 2020).

## Key Finding #3: Evaluation of test automation tools.

The evaluation considered the following codeless and open-source test automation tools:

3.1. Katalon

3.2. UIPath

3.3. Postman/Newman

3.4. Microsoft Playwright

3.5. SikuliX

3.6. Appium

3.7. Selenium

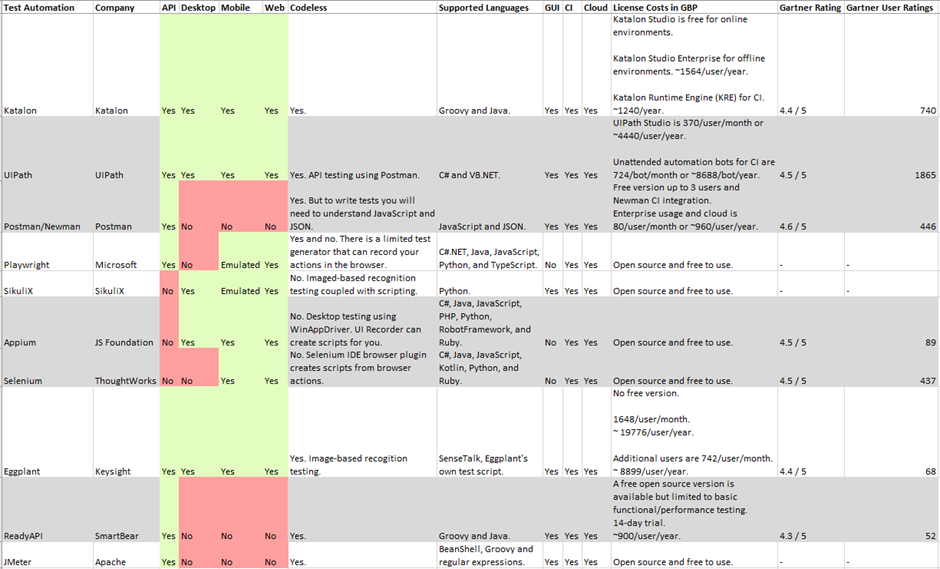
3.8. Keysight Eggplant

3.9. SmartBear ReadyAPI

3.10. Apache JMeter

First we will address each test automation tool in isolation, considering advantages, limitations and then explaining why they suit our test strategy.

Table 1 summarises all of the test automation tools evaluated.



*Table 1 Summary of all test automation tools evaluated.*

**3.1. Katalon**

Katalon is a single Integrated Development Environment (IDE) for API, desktop, mobile and web testing (Altexsoft, 2021; Coyier, 2021; Katalon, 2022).

API testing is built-in, it has a debugger, ready-made code snippets, and response viewer. Open API Specification (OAS) can be imported from related test tools such as Postman or Swagger, and both REST and Simple Object Access Protocol (SOAP) are supported. Desktop testing uses Microsoft’s WinAppDriver enabling you to test most Windows applications. You can visually record your test via actions in the application and you’re able to target buttons and text as required (Altexsoft, 2021).

Web testing under the hood is driven by Selenium and the same tests can be run across all browsers. Tests are created by visually recording your actions in the browser or by writing your own code. XPath, XPath attributes, CSS selectors, images or a mix of location strategies to find buttons and text on web pages. Mobile testing for mobile web, native Apple iOS, and Android, is powered by Appium. Tests can be run locally, using emulators, or real devices (Altexsoft, 2021).

Katalon can be integrated with local CI pipelines using Katalon Runtime Engine (KRE) and on-cloud using Katalon TestOps which has powerful test execution, management, planning, and reporting. Team collaboration is enabled via source control tools such as Git to share changes, Jira for test management, and Microsoft Azure DevOps for cloud CI (Altexsoft, 2021).

A strong feature of Katalon is its easy-to-use, visual reporting. Reports can be exported in many formats including Microsoft Excel and HTML. Tests can be easily managed via Katalon’s clean and user-friendly Graphical User Interface (GUI). It’s simple enough to use without training, although Katalon provides its own academy and there are courses available if required. (Altexsoft, 2021).

Katalon is limited to writing your test scripts in Groovy or Java only and its community is much smaller than mature test automation such as Selenium. Katalon has its own forum and growing community on GitHub so it is gaining traction. Katalon is closed-source software and can only be extended via its own Katalium open-source framework (Altexsoft, 2021).

Other criticisms of Katalon include slow IDE performance and complex mobile testing. There were also reports of problems targeting web page IFrames, however Google has been cutting IFrame JavaScript functions to discourage usage (Altexsoft, 2021; Coyier, 2021).

In summary, Katalon is a good choice for us to unite our team where members are at different levels ranging from beginners to experts. The interface is easy-to-use and it offers us a single tool to perform testing. Global companies that are using Katalon include Ford, Gitlab, Oracle, Samsung, Toyota, and Unilever (Altexsoft, 2021; Katalon, 2022).

**3.2. UIPath**

UIPath is a Robot Process Automation (RPA) tool that enables you to automate most digital processes including but not limited to API, desktop, mobile and web testing. (Kauffman, 2021; Mbaabu, 2021; Sqorus, 2022; Tiwari, 2021; UIPath, 2022).

There are 4 components that make up the UIPath automation. The first component is UIPath Studio which designs automation flows and tests. After visually designing your action sequences in workflows, you convert them into test cases. The second component brings all tests together in the UIPath Test Manager, which as its name implies, provides test management and planning, bug and test reporting, and requirement mapping. The third component is the UIPath Orchestrator which schedules and executes your tests via CI pipelines. The fourth component is the UIPath Robot itself, which carries out the test and then reports back (Tiwari, 2021; UIPath, 2022).

The significant benefit of using UIPath is that anyone can typically create tests and no developer skills are required. Furthermore, the tool adapts to most software and is easily able to target on-screen elements via precise image captures. Once you’ve compiled a set of tests, you can begin running them on a scheduled basis for iterative regression testing (Kauffman, 2021).

Tests can be organised into projects and re-used from your existing test libraries. The tests can be rapidly created and run with or without guidance from the user. UIPath Orchestrator centrally runs the tests, monitors them, and sends you test reports. There is good support for UIPath with an active, international forum and free training available via their own academy (Sqorus, 2022).

There will be a mild learning curve to get up-to-speed with UIPath, so the team will require additional training. One disadvantage to be aware of is that UIPath tests may not work correctly if underlying structures are changed, so form and table tests would fail fast on layout changes. There is a risk that unstable tests may be developed if tests aren’t designed with care when created by image capture (Kauffman, 2021; Mbaabu, 2021).

Overall, UIPath is a good choice for introducing our entire team to test automation, because it greatly simplifies the test creation process. However, tests must be designed with care lest they create unnecessary test maintenance. UIPath may become costly when scaled and it is advised to address testing at lower ends of the testing pyramid, for example doing robust unit testing first (Kauffman, 2021; Mbaabu, 2021; Sqorus, 2022).

**3.3. Postman/Newman**

Postman is a central platform to manage all API flows, testing and OAS documentation. It started as a side project by Abhinav Asthana and was founded in Bangalore, India in 2014. Initially Postman was introduced as a humble Chrome browser extension for debugging APIs but it has grown to become a widely respected tool for API development and testing (AltexSoft, 2019; Charak, 2022; Howard, 2022; Khan, 2021; Postman, 2022).

The Postman GUI allows you to quickly create API tests using pre-prepared JavaScript code snippets. Tests can be shared via Git version control, enabling collaboration. Tests can also be neatly organised into collections, folders, and workspaces. CI integration is easy via the Newman command-line runner by inputting the API collection to test, and then test reports can be quickly generated (AltexSoft, 2019; Howard, 2020).

Using Postman to test large collections of APIs can become unwieldy, because it is better suited for quick manual testing or smaller API collections. It is difficult to re-use test scripts between flows, which is a well-known issue with Postman. Testers will also need to have basic JavaScript skills in order to perform testing (AltexSoft, 2019; Despa, 2020).

Postman supports a test-first approach, enabling us to write tests before features are created. Furthermore, Postman’s interface is user-friendly and while tests can be organised, it may be unsuitable for large API projects (AltexSoft, 2019; Charak, 2022; Postman, 2022).

**3.4. Microsoft Playwright**

Playwright is Microsoft’s open-source API and E2E web test framework introduced in January 2020, built upon the work of Google Puppeteer test library, it has steadily gained popularity (Ahmed, 2021; Heikkilä, 2022; Neova, 2022; Playwright, 2022; Saini, 2021; Schmitt, 2020).

Tests can be written in C#.NET, Java, JavaScript, Python, and TypeScript and the test project changes can be easily shared amongst automation testers via Git source control. Playwright is designed to run on CI pipelines using command-line and it has good reporting features. It is also considered one of the quickest frameworks to run tests, has good browser support including mobile emulation, and has built-in screenshot and video capabilities (Neova, 2022; Playwright, 2022; Saini, 2021).

Limited to API and web testing only, Playwright has an immature community support with most of the useful documentation limited to its own website. Mobile browser testing is also emulated as opposed to testing real devices. Testers will need at minimum to be skilled in a development language such as TypeScript to use Playwright (Ahmed, 2021; Saini, 2021).

Playwright is a fresh and modern test framework for API and E2E web testing, it is not dependent on Selenium, and it’s cloud-ready and designed for running on CI. Test scripts can be effortlessly migrated from existing Puppeteer-style tests, such as legacy Google Protractor. The Playwright framework is lightweight, but testers will still need to have developer skills, and we need to be mindful of limited community support (Ahmed, 2021; Heikkilä, 2022; Playwright, 2022; Saini, 2021).

**3.5. SikuliX**

SikuliX’s name is derived from the Huichol Indian culture, meaning ‘God’s eye’ or the power to see and understand things unknown. It is an open-source automation project, initially started by two students at Massachusetts Institute of Technology (MIT) in 2009, and then was taken over by Raimund Hocke (Actorsfit, 2022; Beniwal, 2017; Nguyen, 2019; OpenCV, 2022; SikuliX, 2021; Yeh *et al.*, 2009).

At the core SikuliX is powered by OpenCV, an open-source computer vision and machine learning library, and is being used by companies such as Google and Microsoft. SikuliX is simple to setup, it is one small Java software that can be installed on multiple platforms. Tests are created using a combination of screenshots and lightweight Java Python code. The tests can then be run via console on a CI pipeline. It is particularly suited for testing software that cannot easily be tested, for example if elements do not have sensible identifiers. Furthermore, it has the ability to read text within documents such as Portable Document Format (PDF) using Tesseract open-source Optical Character Recognition (OCR) (Beniwal, 2017; OpenCV, 2022; SikuliX, 2021; Yeh *et al.*, 2009).

A critical issue of using SikuliX is that it depends on being able to access the screen at all times, so the screen cannot be shared amongst tests. This limits tests to run in serial and time savings from parallel test execution are not possible. Furthermore, image-based test recognition is generally slower than other types of testing. Tests have to be designed by hand, the GUI itself is a bit buggy, and there is no built-in recording capability in the GUI. Typically a good screenshot first must be taken and then it’s combined with a code such as a click function. It is ideal for testing smaller applications, but organising tests for larger applications will not scale. On test reporting, SikuliX has no in-built reporting or Jira integration, so we would need to build custom reporting for test results. The community and documentation for using SikuliX pales in comparison to other test automation, so we would have to be self-sufficient if we encountered problems (Actorsfit, 2022; Beniwal, 2017; Nguyen, 2019).

Overall SikuliX can be used for niche situations where other test automation have failed and could therefore be considered as supporting test automation. It can build smaller scale test suites for applications that aren’t immediately accessible to test. However, tests will fail fast so SikuliX may not be suitable to run iteratively on CI pipelines, but its suitable for longer, periodic testing (Actorsfit, 2022; Beniwal, 2017; Nguyen, 2019; SikuliX, 2021).

**3.6. Appium**

Appium is an open-source test automation written in Node.js to automate mobile applications in Android and iOS using a WebDriver (All About Testing, 2018; Appium, 2018; Test Automation Tools, 2022; The Economic Times, 2022; Saucelabs, 2014).

It received the ‘Open Source Rookie of the Year’ award from Black Duck Software back in 2013 and allows you to use the tools you are most familiar with to write tests, such as C#, Java or other languages. Appium benefits from a loyal community following and its forums are actively used. Furthermore, the same test code developed for mobile Android can also be run on Apple iOS devices and is Selenium compliant. Appium test automation can be run on Android, iOS, and Windows native or hybrid, and the tests can be run in parallel. It does feature a limited GUI recorder and inspector to assist you to create tests (All About Testing, 2018; Test Automation Tools, 2022; The Economic Times, 2022; Saucelabs, 2014).

Appium does not support Android 4.2 and older versions, so you cannot test legacy Android apps, and the tests cannot switch between native and web apps seamlessly, and therefore it can be frustrating to test multiple apps. In addition, Appium has known issues to locate and recognise images so advanced app testing may not be possible. As a tester, you will also need to have basic coding skills to write simple tests using commonly used clicks and gestures. Furthermore, Appium has yet to incorporate Artificial Intelligence (AI) driven testing that can realise efficiencies at scale. Overall, Appium is best fit for basic needs and small teams with limited budget (All About Testing, 2018; Appium, 2018; Test Automation Tools, 2022).

**3.7. Selenium**

Selenium is an open-source web-specific test automation, started in 2004, and it executes tests via its own API. It uses the WebDriver protocol which has a local end (client) that sends commands via your test script to a browser-specific driver. The driver then runs commands against an instance of a web browser (BrowserStack, 2022; Kaur, 2022; Pavan, 2018; Selenium, 2022; SoftwareTestingo, 2020).

It is a powerful, free and open-source test automation due to its convenience, effectiveness and efficiency. It does require time to setup and use but there are extensive information and a plethora of tutorials available on how to use it. Multiple different languages can be used to create and run tests such as C#.NET, Java, JavaScript, and Perl. In addition, it is platform independent and can run on Linux, Mac, Unix and Windows systems. Tests are highly re-usable across systems and they can be run against most web browsers like Chrome, Edge, Firefox, Internet Explorer, Opera, and Safari. Tests can run headless mode without rendering browser graphics for execution efficiency. Moreover, Selenium is easily integrated with other open-source test frameworks such as Behaviour Driven Development (BDD) Cucumber, JBehave Serenity, JUnit, and TestNG. Selenium has good CI and cloud integration with Amazon Web Service (AWS), Azure Devops, Browserstack, Docker, and Jenkins, and can also be run locally. It is a flexible automation that can run tests in parallel to save time and is being actively maintained (BrowserStack, 2022; Kaur, 2022; Pavan, 2018; SoftwareTestingo, 2020).

Since Selenium is an open-source tool it will not have the comfort of direct and professional support. Despite it being light on hardware requirement, Selenium does not function well as a standalone test automation and it relies heavily on other integrations to be able to design and build tests, integrate with CI pipelines and have good test reports. Testers will need to have basic coding skills in a mainstream programming language because testing is not codeless. Another limitation of Selenium is that is only capable to run tests for web browsers and neither desktop nor mobile apps. Furthermore, Selenium is lagging behind the curve because it is not capable to automate on many current application features such as audio, barcode, CAPTCHA security, fingerprints, images, and video. In addition, it has known issues testing frames and popups (Bhattacharjee, 2020; Pavan, 2018; SoftwareTestingo, 2020).

**3.8. Keysight Eggplant**

Eggplant is a black-box GUI test automation tool and was first released in 2002 in Boulder, Colorado, USA before being acquired by UK company Keysight Technologies. Computer vision is the key component powering Eggplant’s test automation. Tests can be created using a hybrid of its own scripting language SenseTalk, image-based recognition, and Selenium. Tests can also be performed from visual models of the application to test (Eggplant, 2022; Goranson, 2021; SoftwareTestingHelp, 2022; Tuteja, 2021).

Tests can be quickly generated using AI and image capture processes, and then tests can be re-used across multiple applications, platforms, and tests. SenseTalk test scripting is relatively simple to use, allowing testers without coding skills to contribute to automated tests. Eggplant’s support team are quick to respond to support calls and will schedule personal video meetings to assist customers (SoftwareTestingHelp, 2022; Tuteja, 2021).

Testers will need some coding skills in order to create re-usable and robust tests, for example a test may need to have a hybrid approach, because pure image recognition tests can be brittle. There are known accuracy issues with OCR technology it uses to recognise text. Furthermore, a good budget will be needed because it is considered an expensive test automation. It has high service costs, a cost per Eggplant test licence, so the cost to run this test automation is higher than tools on the market. Although the test case creation side is quick to setup, its CI integration and reporting aspects are slower to get right (Goranson, 2021; SoftwareTestingHelp, 2022; Tuteja, 2021).

Due to high running cost you should only consider this test automation if you have sufficient budget. Eggplant scales well, allowing everyone to use the same tool to create tests, and upgrades to the test automation since 2021 have not broken tests. However, testers may need some basic coding skills to fully realise Eggplant’s potential (Goranson, 2021; Tuteja, 2021).

**3.9. SmartBear ReadyAPI**

ReadyAPI is an easy-to-use no-code API test automation that enables rapid functional, performance, security and virtualisation testing. The tool is being used by some of the largest companies in the world including FedEx, NASA, Oracle, Pfizer, and Volvo (AltexSoft, 2019; G2, 2022; QA Automation Expert, 2021; ReadyAPI, 2022; Srivastava, 2020).

ReadyAPI supports a wide array of technologies including GraphQL, REST, and SOAP. It has an ever-growing and vast range of integration such as Azure DevOps, GitLab, and Swagger, so iterative API testing can be achieved. Tests can be instantly created from an existing OAS specification such as a Postman or Swagger API project or designed from scratch. The API definitions are separate from tests, so definitions can easily be updated and the same tests can be re-run to validate the APIs. At the heart of the testing, you can visually create test assertions against the APIs without coding skills. Advanced test assertions and data-driven testing are still possible without coding, but you can still script via Groovy if required. API performance and response times can be quickly defined via test assertions. After tests have run via the application or on the CI, there are reports available in Excel, HTML, and PDF formats (QA Automation Expert, 2021; ReadyAPI, 2022; Srivastava, 2022).

Primary criticism of ReadyAPI has focused on its user interface where users reported poor experiences finding correct fields and tabs, as well as poor performance when running intensive tests. There were also reports of ReadyAPI’s GUI being counter-intuitive for first-time users and the ReadyAPI having a lack of usage documentation. Therefore, the user experience could be improved (G2, 2022; Srivastava, 2022).

ReadyAPI will enable our test team to rapidly create API tests from our existing OAS specifications. While the GUI may feel counter-intuitive to new users, it does have attractive in-built performance and security testing that we can instantly apply to our APIs. Fast and iterative feedback via CI and test reports will be possible using ReadyAPI. Furthermore, testers will not require coding skills in order to do the API testing (AltexSoft, 2019; G2, 2022; QA Automation Expert, 2021; ReadyAPI, 2022; Srivastava, 2020).

**3.10. Apache JMeter**

JMeter is an open-source API test automation for functional and performance testing APIs. It supports numerous protocols such as native commands, Java objects, REST, and SOAP (AltexSoft, 2019; Apache JMeter, 2022; BrowserStack, 2020; Toptal, 2018).

JMeter is well-documented and easy to install. It offers a graphical interface to design your tests and will save them into a project in the XML format as a JMeter JMX file. Both functional and performance tests can be created with JMeter. Tests can be run via the GUI or on a CI pipeline via the command-line without requiring a separate runner. JMeter supports graphical charts and reports out of the box. First released in 1998, JMeter is a very mature test automation, has significant community support, and widespread adoption (Apache JMeter, 2022; Toptal, 2018).

Team collaboration on JMeter may not be ideal, because the tests will all be saved into a single XML test project file, which can make test reviews a challenging process. It can be difficult to generate good reports from JMeter if testers are inexperienced. If APIs require pages to process any JavaScript, then this will be difficult to support in JMeter (Apache JMeter, 2022; BrowserStack, 2020).

JMeter is a one-stop shop for API testing, widely supported, and time-tested. Testers will benefit from using its GUI approach to test creation but will still need to be technically informed. Testers will require some basic coding skills to create test assertions, for example using Regular expressions. Support for JMeter is good and there will likely already be a solution if testers are inexperienced to use this tool. It has good support for charts and test reporting, but you will need experience to get the test reporting to work optimally (AltexSoft, 2019; Apache JMeter, 2022; BrowserStack, 2020; Toptal, 2018).

**Comparison of the test automation tools**

We will compare the test automation tools with each other to determine the advantages of using one tool over another.

All of the new test automation will require us to migrate our existing tests over to the new one. Therefore, our ultimate goal should be for good, long-term quality and high test accessibility.

**Katalon, Keysight Eggplant, and UIPath.**

Katalon is preferred for a codeless testing approach, it supports a wide range of testing, and will allow the entire team to contribute to the testing. It is the least expensive in comparison to its rivals Eggplant and UIPath and our main running cost will be on the number of parallel CI runners. If given a suitable budget, then UIPath should be considered as the next choice. However, it will need to be supported by Postman/Newman and API test-driven development (Altexsoft, 2021; Coyier, 2021; Eggplant, 2022; Goranson, 2021; Katalon, 2022; Kauffman, 2021; Mbaabu, 2021; SoftwareTestingHelp, 2022; Sqorus, 2022; Tiwari, 2021; Tuteja, 2021; UIPath, 2022).

**Microsoft Playwright (TypeScript) and Appium/Selenium (C#/Java).**

Microsoft Playwright and TypeScript is preferred for an open-source approach. We must cultivate our team for high-performance test engineering and our main cost will be on the time required for creating new tests via code. Playwright supports API, mobile and web testing, but it will need to be complemented by SikuliX and Python for the desktop testing (Actorsfit, 2022; Ahmed, 2021; Beniwal, 2017; Heikkilä, 2022; Neova, 2022; Nguyen, 2019; OpenCV, 2022; Playwright, 2022; Saini, 2021; Schmitt, 2020; SikuliX, 2021; Yeh et al., 2009).

A second option can be Appium and Selenium with a choice of either C# or Java, however the verbosity of the test cases will increase if switching from using TypeScript. Our testers will need to be experienced developers in order to contribute to the testing (All About Testing, 2018; Appium, 2018; BrowserStack, 2022; Kaur, 2022; Pavan, 2018; Selenium, 2022; SoftwareTestingo, 2020; Test Automation Tools, 2022; The Economic Times, 2022; Saucelabs, 2014).

**What are the major companies using for test automation?**

Ford, Gitlab, Oracle, Samsung, Toyota, and Unilever and over 100,000 companies chose Katalon for their test automation (Katalon, 2022).

DHL, General Electric, Google, HP, NASA, PwC, and Virgin Media are using UIPath for test automation, and they have ~1.5 million members (UIPath, 2022).

Adobe, ADP, Disney, and Sony chose Playwright for test automation (Playwright, 2022).

Expedia, Salesforce, Sauce Labs, Walmart, and Yahoo are using Appium for test automation (Stackshare, 2022).

# Conclusion

Based on the key findings, the following 4 options are suggested in order of preference: In conclusion, our objective was to build a clear picture for choosing codeless test automation to achieve high quality and enhanced test accessibility.

i. Katalon. A codeless test automation with licensed CI integration. Everyone can contribute to testing;

ii. UIPath. An enterprise codeless test automation, used by some of the top companies in the world, but it is expensive. Everyone can contribute to testing;

iii. Playwright. A low-code and open-source test automation. Tests are lightweight and are written in TypeScript. Complemented by SikuliX, we can do desktop testing. Software Developer Engineers in Test (SDETs) can contribute to testing only;

iv. Appium/Selenium. Verbose code and open-source test automation. Tests can be complex and are written in C#.NET or Java. SDETs can contribute to testing only.

The main limitation of the research was the lack of time to examine wider fields of test automation. Other test automation tools can be explored on the G2 Grid for Test Automation. Small, investigative POCs from our test team were used to determine quick insight on new test automation. Furthermore, this research used a mixed approach, striking a balance between sources of literature, POC testing, reviews and team discussion (G2 Grid, 2022).

# References

Actorsfit (2022). *Actorsfit*. Source: <https://blog.actorsfit.com/a?ID=01000-ea29a091-6e51-44bb-b201-81d98bc3e7ea> [Accessed on: 06 July 2022].

Advolodkin, N (2020). *Medium*. Source: <https://nikolay-dev.medium.com/do-we-need-cucumber-for-ui-test-automation-1c902355498f> [Accessed on: 21 June 2022].

Ahmed, M (2021). Source: <https://medium.com/codex/e2e-implementation-of-web-test-automation-using-playwright-b284da4ebe33> [Accessed on: 06 July 2022].

All About Testing (2018). Source: Online, <http://www.allabouttesting.biz/advantages-and-disadvantages-of-appium>, Date: 1 May 2018. [Accessed on: 11 July 2022]

AltexSoft (2019). Source: <https://www.altexsoft.com/blog/api-testing> [Accessed on: 06 July 2022].

AltexSoft (2021). Source: <https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-katalon-studio-automation-testing-tool> [Accessed on: 24 June 2022].

Apache JMeter (2022). *Apache JMeter*. Source: <https://jmeter.apache.org> [Accessed on: 22 July 2022].

Appium (2018). Source: Online, <https://discuss.appium.io/t/advantages-disadvantages-of-using-of-appium/23308>, Date: August 2018. [Accessed on: 11 July 2022]

Applause (2021). Source: <https://www.applause.com/test-automation-a-codeless-approach> [Accessed on: 21 June 2022].

A1qa (2022). Source: <https://www.a1qa.com/blog/scriptless-test-automation-revealing-the-truth-via-3-misconceptions> [Accessed on: 21 June 2022].

Beniwal, A (2017). Source: <https://www.opensourceforu.com/2017/08/software-automation-testing-using-sikuli> [Accessed on: 06 July 2022].

Bhattacharjee, D (2020). Source: Online, *Tutorials Point*, <https://www.tutorialspoint.com/what-are-the-limitations-of-selenium>, Date: 10 June 2020. [Accessed on: 11 July 2022].

Bigelow, S (2021). Source: <https://www.techtarget.com/searchsoftwarequality/CI-CD-pipelines-explained-Everything-you-need-to-know> [Accessed on: 12 August 2022].

Bqurious (2021). Source: <https://bqurious.com/codeless-automated-testing-benefits-and-misconception> [Accessed on: 22 June 2022].

BrowserStack (2020). Source: <https://www.browserstack.com/guide/jmeter-vs-selenium> [Accessed on: 05 August 2022].

BrowserStack (2022). Source: Online, <https://www.browserstack.com/selenium#selenium-web-driver>, Date 2022. [Accessed on: 11 July 2022]

Charak, P (2022). Java Code Geeks. Source: <https://www.javacodegeeks.com/2022/05/understanding-test-driven-development-for-successful-app-releases.html> [Accessed on: 06 July 2022].

Coyier, C (2021). *CSS Tricks*. Source: <https://css-tricks.com/choice-words-about-the-upcoming-deprecation-of-javascript-dialogs> [Accessed on 24 June 2022].

Cucumber.IO (2016). Source: <https://cucumber.io/blog/bdd/getting-started-with-bdd-part-1> [Accessed on: 21 June 2022].

Despa, V (2020). *Medium*. Source: <https://medium.com/apis-with-valentine/how-to-reuse-tests-and-skip-collection-tests-in-postman-da47c50444c8> [Accessed on: 06 July 2022].

Duffy, M (2022). *Automation Testing.* Source: Whitepaper. [Accessed on: 12 August 2022].

Eggplant (2022). *Eggplant Software*. Source: <https://www.eggplantsoftware.com> [Accessed on: 12 July 2022].

FMI (2021). *Future Market Insights*. Source: <https://www.futuremarketinsights.com/reports/codeless-testing-market> [Accessed on: 21 June 2022].

G2 (2022). *G2*. Source: <https://www.g2.com/products/readyapi-test/reviews> [Accessed on: 15 July 2022].

G2 Grid (2022). G2 Grid for Test Automation. Source: <https://www.g2.com/categories/automation-testing?utf8=%E2%9C%93&selected_view=grid&segment=all&attributes%5B219%5D=all#grid> [Accessed on: 12 August 2022].

Goranson, J (2021). *Peerspot*. Source: <https://www.peerspot.com/product_reviews/eggplant-digital-automation-intelligence-review-484494-by-jon-goranson> [Accessed on: 12 July 2022].

Heikkilä, N (2022). Source: <https://nikoheikkila.fi/blog/effortless-end-to-end-testing-with-microsoft-playwright> [Accessed on: 06 July 2022].

Howard, O (2020). *Dzone*. Source: <https://dzone.com/articles/postman-for-api-testing-pros-cons-and-alternative> [Accessed on: 06 July 2022].

Katalon (2022). Source: <https://katalon.com/katalon-studio> [Accessed on: 24 June 2022].

Kauffman, J (2021). *Coveros*. Source: <https://www.coveros.com/robotic-process-automation-pros-and-cons> [Accessed on: 24 June 2022].

Kaur, J. (2022). Source: Online, *DZone*, <https://dzone.com/articles/11-reasons-why-go-for-automation-testing-using-sel>, Date: 23 June 2022. [Accessed on: 11 July 2022]

Khan, A (2021). *Next Big Brand.* Source: <https://www.nextbigbrand.in/postman-journey-of-a-side-project-to-the-leading-api-platform> [Accessed on 06 July 2022].

Knoldus (2022). Source: <https://medium.com/@knoldus/improve-test-quality-using-codeless-solutions-9474f8639c5c> [Accessed on: 22 June 2022].

Martin, D (2020). Source: <https://theqalead.com/move-from-manual-to-automation-testing> [Accessed on: 12 August 2022].

Mbaabu, O (2021). *Section*. Source: <https://www.section.io/engineering-education/introduction-to-uipath-in-rpa> [Accessed on: 22 June 2022].

McDaniel, T (2022). Source: <https://www.virtuoso.qa/post/deprecation-of-protractor> [Accessed on 12 August 2022].

Neova (2022). *Neova Solutions*. Source: <https://www.neovasolutions.com/2022/05/26/api-automation-testing-using-playwright> [Accessed on: 06 July 2022].

Nguyen, T (2019). *LogiGear*. Source: <https://blog.logigear.com/test-automation/12-best-automation-tools-for-desktop-apps-in-2021/#section8> [Accessed on: 06 July 2022].

Nizri, G (2020). *Forbes*. Source: <https://www.forbes.com/sites/forbestechcouncil/2020/10/09/why-no-code-automation-is-now-taking-over-it-operations> [Accessed on: 22 June 2022].

OpenCV (2022). *OpenCV*. Source: <https://opencv.org> [Accessed on: 06 July 2022].

Pavan (2018). Source: Online, *Pavan Testing Tools*, <https://www.pavantestingtools.com/2018/05/10-advantages-and-disadvantages-of.html>, Date: 4 May 2018. [Accessed on: 11 July 2022]

Playwright (2022). *Playwright*. Source: <https://playwright.dev> [Accessed on: 06 July 2022].

PractiTest “State of Testing” (2022). *PractiTest*. Source: <https://www.practitest.com/state-of-testing> [Accessed on: 21 June 2022].

Postman (2022). <https://www.postman.com/company/about-postman> [Accessed on: 06 July 2022].

Pryce, T (2020). *Curiosity Software*. Source: <https://www.curiositysoftware.ie/blog/bdd-gherkin-cucumber> [Accessed on: 21 June 2022].

QA Automation Expert (2021). *QA Automation Expert*. Source: <https://qaautomation.expert/2021/12/27/readyapi-tutorials> [Accessed on: 15 July 2022].

Rana, K (2020). *Techwell*. Source: <https://www.techwell.com/techwell-insights/2020/02/pros-and-cons-codeless-test-automation> [Accessed on: 22 June 2022].

ReadyAPI (2022). SmartBear ReadyAPI. Source: <https://smartbear.com/product/ready-api/overview> [Accessed on: 15 July 2022].

Redwood Logistics (2021). Source: <https://www.redwoodlogistics.com/application-programming-interface-api-transportation> [Accessed on: 12 August 2022].

Rose, S (2021). Source: <https://cucumber.io/blog/bdd/introduction-to-formulation> [Accessed on: 21 June 2022].