OPEN ACCESS

Volume: 4

Issue: 2

Month: April

Year: 2025

ISSN: 2583-7117

Published: 08.04.2025

Citation:

Mrs. Elavarasi Kesavan "A Comprehensive Review of Automated Software Testing Tools and Techniques" International Journal of Innovations in Science Engineering and Management, vol. 4, no. 2, 2025, pp. 14–20.

DOI:

10.69968/ijisem.2025v4i214-20

A Comprehensive Review of Automated Software Testing Tools and Techniques

Mrs. Elavarasi Kesavan[1]

[1]*Full Stack QA Architect, Cognizant*

Abstract

Automation testing is often needed in the software testing process to improve test results and save time and money. Automation testing is very useful for evaluating web applications based on load, stress, security, vulnerability, and performance. The many studies on automated software testing methods and tools are reviewed in this article. This comprehensive review highlights that no single automated testing tool excels in all aspects; the selection depends on project-specific criteria such as cost, usability, support, and test type. Tools like Selenium and WATIR offer flexibility and cost-efficiency for web applications, while UFT/QTP supports both web and desktop testing but at a higher cost. The rise of AI-powered, self-repairing frameworks signals a shift toward more adaptive testing solutions. Additionally, parallel execution methods significantly enhance regression testing efficiency. Ultimately, selecting the right tool and execution strategy is crucial to minimizing time and cost in software development.

Keywords; Software testing process, Selenium, WATIR, Automation software testing tools and techniques, Functional testing tools, Code coverage tools, Automated software engineering (ASE), etc.

INTRODUCTION

As computer software use has grown, so too has awareness of the significance of software management and quality. One aspect of software engineering that has to do with the final product's quality is software testing. Software quality is ensured by using the test to confirm, validate, and estimate the dependability of software products. For when software systems' quality deteriorates, catastrophic outcomes might happen [1]. Due to all of these problems, since the late 1970s, more time and personnel have been devoted to the software testing process. As a result, software testing has emerged as one of the most challenging and necessary procedures for businesses, organisations, and academics. This is because software products are used in a broad variety of applications, from everyday life to mission-critical systems, depending on the always growing demands [2]. Software testing involves various factors, ranging from front-end to back-end testing. Numerous functional and non-functional software testing methodologies exist. Acceptance, system, integration, and unit testing are all examples of functional testing methodologies that should be used sequentially [3]. The following approaches pertain to the operational elements of software and are considered non-functional testing techniques: compatibility, security, usability, and performance testing [4]. The fundamentals of software testing include determining whether the chosen test methods are appropriate for a given software, how to conduct a scope test, which document test procedures to follow, what test types and methods to use, what software requirements there are, which technique to use for verification and validation, and so on [5].

In order to achieve notable gains in productivity and quality, Automated Software Engineering (ASE) uses software tools and methodologies to automate the processes of software system analysis, design, implementation, testing, and maintenance. Large and ultra-large software businesses have tackled these issues by using ASE tools and methodologies [6].

However, because of various limits and settings, they may not work well for small and medium-sized businesses (SMEs). Big or ultra-large software organisations often have greater resources, so they can spend money on R&D, training, and other things. It might be more difficult for SMEs to implement sophisticated or even modern ASE technologies and processes since they often lack the means to do so [7]. In contrast to big or ultra-big businesses, they could still use procedures from their outdated software or have differing requirements from their development teams and target consumers. Automated tool assistance is also essential for the research community to better assist software engineers in all sizes of software firms (e.g., decreasing rework, enhancing timely delivery, and controlling accidental complexity) [7], [8].

Software testing

Software testing is the process of assessing and confirming that an application or software product performs as intended. Performance improvement and bug prevention are two advantages of thorough testing. The most successful software testing nowadays is continuous, meaning that testing begins during design, continues throughout software development, and even happens during production deployment [8]. Continuous testing eliminates the need for organisations to wait for the deployment of all components before beginning testing. Test philosophies that have gained popularity lately in the software world include shift-left, which brings testing closer to design, and shift-right, which involves end users doing validation. Automating every part of testing is crucial to enabling the necessary speed of delivery once your test strategy and management objectives are clear [9].

Automation software testing

One essential method in software development is automated software testing, in which testers run test cases automatically using specialised tools. For repeated, resource-intensive operations that are difficult to do by hand, this method saves time and effort. Automation speeds up time to market, lowers related costs, and improves software quality by simplifying the testing process. Numerous automation technologies are capable of managing a wide range of desktop, mobile, and online applications. Any software-driven company hoping to produce high-quality products more rapidly must use automated software testing in today's fast-paced development settings [10].

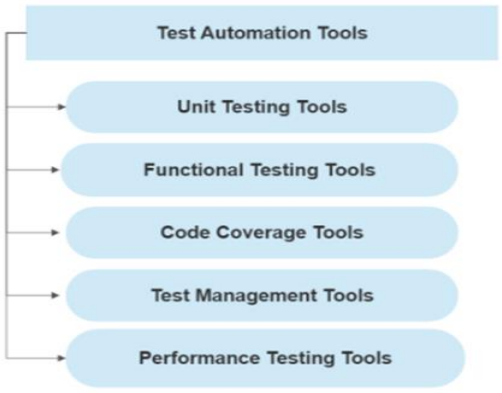
Importance of automated software testing

For many strong reasons, automated testing is a crucial strategy. The first benefit is that it greatly increases productivity by automating repeated testing operations, which saves time and money compared to manual testing. This increase in efficiency frees up development teams to concentrate more on innovation and providing consumers with value [11]. Furthermore, by lowering the possibility of human error—a frequent problem with manual testing—automated testing increases software dependability by producing consistent and reproducible test findings. Faster feedback loops are another benefit of automated testing, which helps teams find and repair defects early in the development cycle and reduces the likelihood of expensive solutions. All things considered, automated testing is essential in today's fast-paced software development environment as it helps guarantee the quality, dependability, and timely delivery of software products [12].

Automated Software Testing Tools

As seen in Fig. 1, software testing automation technologies may be used for performance testing, code coverage, functional testing, unit testing, and test management. Continuous Integration is a typical engineering idea that is incorporated into automated testing (CI). A software development technique called continuous integration entails incorporating code modifications from many developers on a regular basis into a common repository [13]. Since automated tests are run automatically whenever new code is merged, they are an essential part of continuous integration (CI). By doing this, you can make sure that any new modifications won't cause bugs or interfere with already-existing functionality. Early in the development cycle, problems may be found and resolved since developers can quickly assess the codebase's health by including automated tests into the integration process. Collaboration is encouraged, integration issues are decreased, and software engineering projects retain a high-quality codebase thanks to continuous integration [14].

Figure 1 Automated Software Testing Tools [14]



Unit Module Testing Tools: Core code pieces are validated by this kind of testing tool. These technologies, which are easily included into development platforms like NetBeans, are crucial for automating testing processes. Developers may use programming languages to automatically run test cases by using automated testing. In addition to looking over each line of code in the program, unit testing tools are essential for confirming the exact execution of certain units or methods and guaranteeing their proper working by examining the code structure and following best practices. JUnit, PHPUnit, NUnit, and JMockit are a few of the unit testing frameworks.

Function Testing Tools: This kind of testing is done to make sure the program works according to the users' needs. We make advantage of the available functional testing tools while doing functional testing. By providing functions with input and then comparing their actual output with the anticipated result for the particular test case, functional testing tools are used to evaluate how well functions execute. To evaluate how well a software system complies with the requirements, functional testing technologies are used [15]. There is nothing to rewrite in the user's text. Selenium, HP QuickTest Professional, TestComplete, Ranorex, Watir, Tricentis Tosca Testsuite, and Test Studio are a few examples of functional testing technologies that are often used in software testing.

Code Coverage Tools: Which parts of the software code have been successfully tested by the different testing tools is determined by these automated tests. It keeps track of the quantity of lines, statements, or code blocks that have undergone verification by automated test suites. To evaluate the effectiveness of the Quality Assurance (QA) initiatives that have been carried out, this metric is crucial.

Software for Managing Tests: Test plans (including test cases, test plans, test strategies, test results, test reports, etc.) may be automated with the use of these tools, which can help teams manage projects more effectively by providing a searchable and maintainable placeholder for test operations. Each of the several test management options has a unique set of tools and techniques for overseeing the testing process. On the other hand, they often provide the chance to streamline the testing procedure, enable immediate access to data analysis, and promote collaboration across different project teams.

Performance Testing Tools: Techniques for performance testing are implemented during the testing process. The purpose of these evaluations is to assess the program's responsiveness and stability in a diverse array of scenarios and demand levels. In addition, it can be employed to evaluate, quantify, validate, or verify other aspects of software quality, such as dependability, scalability, and resource utilisation. These techniques are effective in assessing the performance of software or components in terms of resource use, throughput, and stimulus-response time in accordance with a specific set of performance standards [16]. In addition to endurance and surge testing, performance testing categories that offer additional benefits include configuration, isolation, Internet testing, breakpoint testing, and immersion testing.

Automated software testing techniques

Unit Testing: Software development is fundamentally characterised by unit testing, which is the process of verifying the accuracy of individual units or components within a software system. Unit testing's primary objective is to guarantee that each isolated unit, which may exist as a function, method, or module, functions as intended. In order to verify both anticipated and unexpected inputs and outputs, developers define test cases that encompass a wide range of scenarios. Unit testing plays a crucial role in the early identification of defects during the development process, which in turn enables the more efficient and cost-effective resolution of bugs [17]. Developers are granted assurance that their modifications to the code will not inadvertently disrupt existing functionality by these tests, which serve as a safety net.

Integration Testing: Integration testing is a complementary approach to unit testing that involves the analysis of the interactions between various units or components of a software system. It is the primary goal to identify potential issues that may arise during the integration of these components and to guarantee their seamless operation. Integrity testing is instrumental in the identification of issues such as communication errors, interface discrepancies, and data flow issues [18]. Two primary methodologies for integration testing are top-down and bottom-up. In a top-down approach, testing commences with the highest-level modules and progresses into the lower-level ones. On the other hand, a bottom-up approach commences with the testing of lower-level modules, which are subsequently integrated and tested.

System Testing: A comprehensive testing phase that assesses the entire software system as a whole is known as system testing. Whether the specified requirements are met by the entire application, which includes all integrated components, is verified. A variety of test scenarios, such as

functional, performance, security, and usability testing, are included in system testing. The goal is to guarantee that the software operates in accordance with the set criteria and performs as anticipated in a variety of environments. A comprehensive evaluation of the software's suitability for deployment is frequently conducted during the final testing phase prior to its release to users, known as system testing.

Load Testing: One kind of performance testing that evaluates how a software system behaves under expected load scenarios is called load testing. The objective is to guarantee that the application can manage the anticipated workload and user traffic without experiencing any performance deterioration. The purpose of load testing is to assess the system's scalability and find any bottlenecks by simulating actual use situations, including times of peak demand. By doing this, businesses can guarantee a satisfying user experience under a range of load scenarios, improve responsiveness, and maximise system performance.

Error Testing: A software system's resilience and error-handling skills are evaluated by purposefully introducing mistakes, erroneous inputs, or unexpected situations. This technique is called error testing, often referred to as negative testing. The objective is to determine how effectively the application recognises, reports, and recovers from unexpected or incorrect circumstances. Error testing makes software more dependable and user-friendly by ensuring that it responds gracefully to unexpected inputs or circumstances. In order to find vulnerabilities, stop unplanned system failures, and enhance the overall quality of the program, this kind of testing is essential.

Test Automation: Test automation is the practice of replacing manual testing efforts with automated procedures by automating the execution of test cases using specialised tools and scripts. Enhancing the testing process's accuracy, efficiency, and repeatability is the main objective of test automation. Compared to manual testing, automated tests may be conducted more often and reliably, giving developers quick feedback and facilitating the early identification of flaws. In regression testing, where previously verified features are retested to make sure that new modifications haven't caused problems, test automation is very helpful. It speeds up software development overall, facilitates pipelines for continuous integration and continuous delivery (CI/CD), and quickens the testing cycle.

LITERATURE REVIEW

(Kumar & Rodda, 2025) [19] Evaluating automated test failure detection and repair tools in software test automation was the primary goal of this work. Through a comparison of the previously evaluated sources, the evaluation highlights the advantages and disadvantages of each, as well as potential areas of use. Additionally, it offers recommendations for suitable applications in a wide range of testing scenarios and settings. The review also begins with the shortcomings and difficulties that the state-of-the-art methods have encountered, and it provides a forecast for the future of automated test failure detection and repair research and development. A review of the most widely used tools is included at the conclusion of this article, along with the opinions offered about current and future artificial intelligence for robotic testing.

(Hussein Mohammed Al et al., 2023) [14] Testing is a crucial step in identifying problems and evaluating the product's quality, and it plays a significant role in the software development lifecycle. figuring out software testing techniques that save time and money. The study looks at the advantages and disadvantages of automated testing, how it simplifies software quality evaluation, and how it saves time in comparison to manual testing. With automated testing solutions, software application testing is made easier, tailored to specific testing scenarios, and accomplished with success.

(Samli & ORMAN, 2023) [20] Various producers have devised a variety of web-based automated instruments with varying properties. Deciding which web-based automated application is most suitable for a particular testing process is frequently a challenging task. The process of conducting trials and selecting the appropriate automated tool is occasionally impossible due to the high cost and licensing requirements of many of these tools. By focussing on a limited number of instruments and comparison criteria, certain studies in the literature attempt to address this issue. Still, the comparison of automated tools is a critical matter that necessitates further investigation. For the first time in the literature, this paper compares 14 web-based automated tools based on 20 distinct criteria. The results of this comprehensive review are reported.

(EROL & SENAN, 2022) [5] Compare and evaluate various test automation tools based on factors such as cost, performance, and usability in order to address this resource deficit. The most popular software test automation tools have been analysed for this purpose by conducting experiments on computers with varying hardware specifications and websites. The operating conditions of software testing instruments are also contrasted and analysed under the same conditions. This approach is

intended to provide guidance for assessing the capabilities and properties of the testing instruments that have been examined. In the software development process of websites, this study is expected to serve as a valuable resource for minimising testing costs and time.

(Malik & Mehta, 2022) [21] The automation and development approaches are clearly demonstrated in this paper. The research findings unequivocally demonstrate that the usability, security, efficiency, and reliability of current and future software programs are on the brink of transformation as a result of the new strategy of automating software testing realms. Testing is a critical component of the software development process. This paper addresses a critical and significant matter in the field of software testing. Testing may be implemented manually or automatically. Each of these methodologies has its own set of advantages and disadvantages. The purpose of this paper is to conduct automation testing using the Selenium software testing tool. While the tester is inputting data into a web application interface, this web testing instrument automatically records test cases in the background.

Table 2Adavantages and disadvantages of automated software testing [21]

Advantages	Disadvantages
Enhances the quick finding encounters of bugs with heightened accuracy	Disadvantages Inherent knowledge of the tool required
Saves time and energy since the process is quite efficient	Considerable time is lost in choosing the right approach and tool
The test script can be comfortably repeated	Initial cost of acquiring a test tool is high
Improves the quality of the testing process and software accuracy	If playback approach is the option, test maintenance is quite a burden
Increased coverage owing to the multiple testing tools for parallel testing encounters	High levels of proficiency required to creat the scripts needed for the test.

(Alferidah & Ahmed, 2020) [22] sought to identify instruments that could enhance the reliability, efficacy, and quality of the software systems. To ensure that a comprehensive system test is conducted without any errors that could have resulted in financial losses, automated software testing tools are the solution. The majority of automated software testing tools enable the tester to repeat and utilise the final product, boost reliability and performance, and reduce the necessary time. In this paper, the most effective and efficient automated software testing instruments are described.

(Gamido & Gamido, 2019) [23] Because it confirms if the system satisfies user needs and specifications, software testing is regarded as one of the most crucial steps in the software development process. Software testing may be done in two ways: manually and automatically. Software testers may easily automate the software testing process via automated testing, which makes it more efficient in terms of time, cost, and usability. Numerous automated testing tools, both open source and commercial, are available. In order to assist customers in choosing the best software testing tool for their needs, this article compares the characteristics of open source and commercial testing solutions.

(M. A. Umar & C. Zhanfang, 2019) [24] Two critical components of a successful and effective software testing project are the use of appropriate testing methods and the appropriate test automation tools/framework. Utilising a single testing methodology will not suffice to evaluate software and guarantee its quality; rather, a combination of suitable testing methodologies is frequently necessary. The initial step in attaining successful and efficient software testing is to be familiar with the various testing methods and tools/frameworks. A comprehensive examination of test automation frameworks and tools is presented in this article. Initially, the description of automated testing and its classifications is provided, followed by an explanation of the numerous test automation frameworks. A succinct explanation and comparison of several of the most frequently employed automation tools are provided.

(Sutapa et al., 2019) [25] For the benefit of other researchers working in this area, an overview of the automated testing methodology is provided. The results of the review demonstrate that the automated testing strategy is appropriate for improving regression testing with several tenable tool alternatives, such as Robot-framework, SAHI, and Selenium. Additionally, this article discusses a few tenable execution technique possibilities. This review also comes to the conclusion that one potential option for carrying out the most effective regression testing procedure is the parallel execution approach.

(Sneha & Malle, 2017) [26] The only way to determine the quality of any program is via testing (software testing). In the testing process, automation testing has had an influence. The majority of software testing is now done using automated tools, which reduces the number of humans working with the program and the faults that may be overlooked by the tester. Test cases are included in automation testing, which facilitates the process of capturing and storing various situations. Thus, the practice of software

automation testing is essential to the success of software testing. The goal of this research is to compare manual and automated testing, as well as to understand the many forms of software testing and the tools and methods used in software testing.

CONCLUSION

This comprehensive review of automated software testing tools and techniques highlights the significance of selecting the right tool based on project-specific needs. The evaluation of tools such as Selenium, QTP/UFT, TestComplete, Ranorex, WATIR, Sahi, and SoapUI reveals that no single tool is universally superior. Open-source tools like Selenium are cost-effective and support cross-browser and cross-platform testing but are limited to web applications. Licensed tools like QTP/UFT offer broader application testing capabilities, including web and desktop, along with advanced features like object identification and ALM integration, though at a high cost. WATIR provides flexibility but lacks record and playback features. Emerging AI-based testing frameworks show promise in addressing the challenges of script maintenance for dynamic web applications. Moreover, regression testing can benefit greatly from automation, especially when using parallel execution methods to reduce testing time. This study provides valuable insights for developers and testers in choosing the appropriate automation tool, aiming to optimize testing efficiency, reduce costs, and accelerate the software development lifecycle.

REFERENCES

[1] Y. Wang, M. V. Mäntylä, Z. Liu, and J. Markkula, “Test automation maturity improves product quality—Quantitative study of open source projects using continuous integration,” J. Syst. Softw., vol. 188, p. 111259, 2022, doi: 10.1016/j.jss.2022.111259.

[2] A. S. Verma, A. Choudhary, and S. Tiwari, “Software Test Case Generation Tools and Techniques: A Review,” Int. J. Math. Eng. Manag. Sci., vol. 8, no. 2, pp. 293–315, 2023, doi: 10.33889/ijmems.2023.8.2.018.

[3] D. Amalfitano, S. Faralli, J. C. R. Hauck, S. Matalonga, and D. Distante, “Artificial Intelligence Applied to Software Testing: A Tertiary Study,” ACM Comput. Surv., vol. 56, no. 3, 2023, doi: 10.1145/3616372.

[4] N. Islam, “A Comparative Study of Automated Software Testing Tools,” Culminating Proj. Comput. Sci. Inf. Technol., vol. 9, no. 19, pp. 9211–9219, 2016.

[5] E. EROL and S. SENAN, “A Comparative Study for Evaluating Automated Software Testing Tools,” Bilişim Teknol. Derg., vol. 15, no. 3, pp. 301–316, 2022, doi: 10.17671/gazibtd.1057380.

[6] N. Anwar and S. Kar, “Review Paper on Various Software Testing Techniques & Strategies,” Glob. J. Comput. Sci. Technol., vol. 19, no. 2, pp. 43–49, 2019, doi: 10.34257/gjcstcvol19is2pg43.

[7] C. Ragkhitwetsagul, J. Krinke, M. Choetkiertikul, T. Sunetnanta, and F. Sarro, Adoption of automated software engineering tools and techniques in Thailand, vol. 29, no. 4. 2024. doi: 10.1007/s10664-024-10472-6.

[8] J. J. Li, A. Ulrich, X. Bai, and A. Bertolino, “Advances in test automation for software with special focus on artificial intelligence and machine learning,” Softw. Qual. J., vol. 28, no. 1, pp. 245–248, 2020, doi: 10.1007/s11219-019-09472-3.

[9] Arun Kumar Arumugam, “Software Testing Techniques New Trends,” Int. J. Eng. Res., vol. V8, no. 12, pp. 708–713, 2019, doi: 10.17577/ijertv8is120318.

[10] G. Murazvu, S. Parkinson, S. Khan, N. Liu, and G. Allen, “A Survey on Factors Preventing the Adoption of Automated Software Testing: A Principal Component Analysis Approach,” Software, vol. 3, no. 1, pp. 1–27, 2024, doi: 10.3390/software3010001.

[11] D. S. Battina, “Artificial Intelligence in Software Test Automation: A Systematic Literature Review,” JETIR, vol. 6, no. 12, pp. 181–192, 2019, doi: 10.5220/0009417801810192.

[12] N. G. Berihun, C. Dongmo, and J. A. Van der Poll, “The Applicability of Automated Testing Frameworks for Mobile Application Testing: A Systematic Literature Review,” Computers, vol. 12, no. 5, 2023, doi: 10.3390/computers12050097.

[13] M. Hanna, A. Elsayed, and M.-S. M., “Automated Software Testing Frameworks: A Review,” Int. J. Comput. Appl., vol. 179, no. 46, pp. 22–28, 2018, doi: 10.5120/ijca2018917171.

[14] Hussein Mohammed Al, M. Y. Hamza, and T. A. Rashid, “A Comprehensive Study on Automated Testing with The Software Lifecycle,” J. Duhok Univ., vol. 26, no. 2, pp. 613–620, 2023, doi: 10.26682/csjuod.2023.26.2.55.

[15] F. N. Musthafa, S. Mansur, and A. Wibawanto, “Automated Software Testing on Mobile Applications: A Review with Special Focus on

Android Platform," Int. Conf. Adv. ICT Emerg. Reg., vol. 36, no. 3, pp. 1–4, 2020, doi: 10.1145/1968587.1968601.

[16] Ashritha S and Padmashree T, "Machine Learning for Automation Software Testing Challenges, Use Cases Advantages & Disadvantages," Int. J. Innov. Sci. Res. Technol., vol. 5, no. 9, 2020, [Online]. Available: www.ijisrt.com

[17] N. U. Ansari and P. Richhariya, "Deep Hybrid Intelligence : CNN-LSTM for Accurate Software Bug Prediction," Int. J. Innov. Sci. Eng. Manag., pp. 26–33, 2024, doi: 10.69968//ijisem.2024v3i426-33.

[18] A. Banga and R. Arora, "Advancements in Automation Testing Optimization : A Comprehensive Review of Recent Techniques and Trends," Int. J. Sci. Res. Sci. Eng. Technol., pp. 344–355, 2024.

[19] N. H. Kumar and S. Rodda, "Comparative Review on Automated Test Failure Detection and Healing Tools," SSRG Int. J. Electr. Electron. Eng., vol. 12, no. 2, pp. 113–123, 2025, doi: 10.14445/23488379/IJEEE-V12I2P113.

[20] R. Samli and Z. ORMAN, "A Comprehensive Overview of Web-Based Automated Testing Tools," İleri Mühendislik Çalışmaları ve Teknol. Dergisi, vol. 4, no. 1, pp. 13–28, 2023.

[21] A. Malik and A. Mehta, "Automation Testing-a Review," Int. Res. J. Mod. Eng. Technol. Sci., no. 06, pp. 2582–5208, 2022, [Online]. Available: www.irjmets.com

[22] S. K. Alferidah and S. Ahmed, "Automated Software Testing Tools," Int. Conf. Comput. Inf. Technol. ICCIT 2020, no. September, 2020, doi: 10.1109/ICCIT-144147971.2020.9213735.

[23] H. V. Gamido and M. V. Gamido, "Comparative review of the features of automated software testing tools," Int. J. Electr. Comput. Eng., vol. 9, no. 5, pp. 4473–4478, 2019, doi: 10.11591/ijece.v9i5.pp4473-4478.

[24] M. A. Umar and C. Zhanfang, "A Study of Automated Software Testing: Automation Tools and Frameworks," Int. J. Comput. Sci. Eng., vol. 8, no. 06, pp. 217–225, 2019, doi: 10.5281/zenodo.3924795.

[25] F. A. K. P. G. Sutapa, S. S. Kusumawardani, and A. E. Permanasari, "A Review of Automated Testing Approach for Software Regression Testing," IOP Conf. Ser. Mater. Sci. Eng., vol. 846, no. 1, 2019, doi: 10.1088/1757-899X/846/1/012042.

[26] K. Sneha and G. M. Malle, "Research on Software Testing Techniques and Software Automation Testing Tools," Int. Conf. Energy, Commun. Data Anal. Soft Comput., pp. 77–81, 2017.