Impact of Automation on Quality Assurance Testing: A Comparative Analysis of Manual vs. Automated QA Processes

Amit Bhanushali

Independent Researcher, West Virginia University, WV, USA,
ORCID: 0009-0005-3358-1299

[1]*Received: 29 September 2023; Accepted: 21 October 2023; Published: 24 October 2023*

ABSTRACT

The practice of assessing and guaranteeing the high quality of software or systems via the use of automated tools and procedures is known as automation quality assurance. It includes the use of software scripts, test automation frameworks, and other technologies to conduct testing and verification duties in a manner that is more efficient and consistent, with the end goal of eventually improving the overall quality of the product or system that is being tested. This study aims to analyze the effects of automation on Quality Assurance (QA) testing by comparing manual and automated QA processes. As organizations continue to embrace automation in their software development workflows, it is crucial to comprehend the impact of this transformation on the quality assurance domain. In view of Top to bottom experimental exploration, this paper tries to give a careful assessment of the qualifications, advantages, and hindrances among human and automated quality assurance testing. Additionally, it delves into the changing responsibilities of QA professionals in the era of automation, highlighting the importance of acquiring new skills and being adaptable. This research adds to the ongoing discussion about QA testing in the digital era and provides valuable insights for organizations looking to improve their QA processes. In order to provide thorough software quality assurance, this study's findings emphasise the importance of implementing a balanced strategy that combines the benefits of both manual and automated QA testing.

Keywords: *Automation; Automated; Manual; Quality; Assurance; Testing*

INTRODUCTION

The Product testing guarantees the quality and trustworthiness of the completed item and is a pivotal move toward the product improvement process. Software testing is done to find defects and errors in the programme and make sure it satisfies both the functional and non-functional needs of the end users. The two most common techniques for evaluating software are manual testing and automated testing. To check the functionality of the product, manual testing comprises evaluators running test cases manually without the aid of software tools. On the other side, automation testing involves using software tools to carry out tests automatically and without human involvement. The testing approach chosen has a considerable impact on the efficacy and effectiveness of the software testing process. For many years, manual testing has been the accepted method for testing software, and it is still used in much software development Projects today. With the advancement of technology, however, computerized testing has gained popularity in recent years. Automatic testing increases testing efficacy, improves test coverage, and reduces testing costs and duration**. [1]**

In the past few decades, software development has undergone a rapid evolution, with an increasing number of functionalities, interactions, and system dependencies. **[2]** As software system complexity increases, ensuring their quality has become increasingly difficult. To track down blemishes and guarantee programming quality, programming testing is a fundamental stage in the product improvement process. Because of developing rivalry and market interest, the product improvement industry is feeling the squeeze to make great programming frameworks inside close courses of events. Therefore, it is more crucial than ever to improve the effectiveness and efficiency of software testing to achieve a shorter time to market. The traditional approach to software testing has been manual testing. However, it can be time-consuming, costly, and prone to error, particularly for large and complex software systems. Considering the

[1] *How to cite the article:* Bhanushali A (October, 2023); Evolving Trends in Quality Assurance Testing: A Comprehensive Review of Methodologies and Tools; *International Journal of Advances in Engineering Research;* Vol 26, Issue 4; 39-54

numerous interactions and dependencies between various software system components, the effort required for manual testing becomes even more pronounced. Therefore, many software development teams are investigating automated testing methods as an alternative to manual testing in order to circumvent its limitations. Automated testing uses scripts and tools to automate the testing process, reducing the time and effort necessary to execute test cases. **[3]** By automating testing, developers can expedite the testing cycle while providing an efficient means of detecting defects and ensuring software quality. Although automated testing has been extensively adopted by many software development teams, the software engineering community continues to debate its effectiveness in enhancing software quality. Some researchers contend that automated testing can enhance software quality by reducing bugs and expanding test coverage. Others argue that it may not always be effective in identifying defects or even in introducing new ones.

The process of running a programme in order to find faults is known as software testing [4]. Software testing enhances the integrity of software that is still being developed [5]. Testing is a powerful tool for raising standards. The two testing approaches are manual testing and automated testing. Manual testing is additionally alluded to as static testing. The inspector directs the assessment. Dynamic testing is one more name for automated testing [6]. Because of this speed increase, it was expected to speed up and run tests more algorithmically at various areas. Software testing automations and manual tests have quickly entered the market in response to this need. Typically, manual testing is time-consuming, expensive, and disorganized. The following are some of the primary causes for the development of automation.

⇒ Improved Test Coverage

⇒ Simulated User Environment

⇒ Greater Return on Investment: Saves Time and Money

⇒ Volume and Concurrent Execution

⇒ Early Bug Detection **[7]**

Unlike manual test applications, automation test applications are different. Automation testing are not always appropriate, in contrast to manual tests. The main uses for automated testing are load and performance tests, smoke tests, static and repetitive tests, and regression tests (as well as data-driven tests) [8]. These assessment types depend on quantifiable data. Testing utilizing automation can be utilized to evaluate both practical and non-useful test types. The use of test automation devices has extraordinarily progressed.

Automation methods, however, necessitate manual testing, contrary to popular assumption. Testing is an expensive part of the software development process. A good way to cut these costs has been suggested: test automation. Automation testing calls for manual test sessions [9]. Prior to automating any circumstance, the manual operation of the issue at hand must be determined. Because of this, manual tests and automated tests cannot be separated. As it relates to this, the following Increasing the degree of test automation and risk reduction is a strategy to support manual testing as an alternative to adding more staff during testing. One of the most important components of automation is choosing the right testing framework and instrument. These difficulties have been the subject of study and early investigations, allowing for better automated process management. With regards to this subject, assessing test automation arrangements is a tedious system that requires careful exploration. **[10].**

FRAMEWORKS FOR AUTOMATED TESTING

A bunch of guidelines or rules utilized in the turn of events and plan of experiments is alluded to as a testing structure. A structure is comprised of a bunch of systems and hardware that are planned to assist QA experts with directing tests all the more rapidly. The principles might address different subjects, including coding shows, ways to deal with taking care of test information, object storehouses, strategies for putting away experimental outcomes, and bearings for exploring to different assets. In spite of the fact that it isn't needed, involving an organized system for prearranging or recording tests could give a few additional benefits that could somehow slip through the cracks. [11]

Implementing a test automation framework has a number of advantages. First and foremost, a test automation framework aids in enhancing the efficacy and efficiency of the testing procedure. Testers can concentrate on more important areas of testing, such as exploratory testing and result analysis, by automating repetitive and time-consuming operations. A group's testing pace and effectiveness, test rightness, and test support costs can be in every way improved by executing an automated testing structure. It can also assist in reducing testing-related dangers. They are necessary for an effective automated testing procedure for a number of reasons. **[12]**

⇒ Enhanced efficiency of testing

⇒ Reduced maintenance expenses

⇒ The need for manual intervention is kept to a minimum.

⇒ Achieving maximum test coverage is essential for ensuring the thoroughness and effectiveness of our testing efforts.

The concept of code reusability refers to the ability to use existing code in different parts of a program or in different programs altogether.

Systems for automated testing arrive in various structures. The automation of programming testing processes is made simpler by these structures. There are six well known classes of test automation systems, each with an unmistakable plan and a scope of advantages and disadvantages. It is fundamental to pick the structure that best meets your goals while thinking up a test system.

1) Linear Automation System

This original testing structure, the most essential of its sort, can be utilized to test the UI (UI) of a web application. To direct the test consecutively all through this stage, the analyzer will record each step, for example, route, client information, or designated spots, and afterward play the content back. The analyzer doesn't have to compose code during this stage to make functionalities. It is otherwise called a record-and-playback system in certain areas. **[13]**

2) Framework for Modular-Based Testing

This sort of test automation system separates the product under test into different exceptional capabilities, modules, or segments prior to testing every one of these parts freely. Each component in a hierarchical method has a separate test script, which is subsequently combined with test scripts from other components to generate larger tests. Soon, a range of test scenarios will start to be represented by these increasingly thorough sets of tests. The crucial step in achieving modularity is breaking down the functionality and then recombining the modules [14]. The functional decomposition framework, which is the modular automation framework's other name, is used in specific contexts. **[15].**

3) Framework for testing library architecture

This structure, which additionally has specific additional advantages, is based on the secluded system. It perceived related errands or capabilities inside the product that should be tried and gathered them by capability, rather than parting the developer under test into a few free modules or units to be tried in confinement with their own test scripts. This made it conceivable to rapidly test the undertakings or administrations more. Eventually, the software engineer is separated into its constituent areas relying upon the common goals or capabilities. The test contents can get to these schedules at whatever point they see fit since they are kept in a library [13]. Since it has a library of normal capabilities that various test contents can use, this system offers a more elevated level of reusability than the Measured Based Structure. Conversely, there is no identical library in the Measured Based System.

4) Framework that is data-driven

The product testing industry is notable for its utilization of the information driven test structure. The test information is isolated from the content rationale and put away remotely to an outer information source in this approach. Text documents, Succeed bookkeeping sheets, CSV records, SQL tables, and ODBC stores are a couple of instances of these information sources. An outside information source is associated with test contents, and they are carefully guided perused and populate the essential information at the proper times [13]. This system isolates the test information from the test script, in contrast to straight automation, particular based design, and library engineering. Thus, without changing the test script each time, analyzers can test a similar programming capability or element on numerous occasions utilizing various arrangements of test information.

5) A Framework Driven by Keywords

This system embraces a technique that is basically the same as that of an information driven structure in that test information and content rationale are both isolated; notwithstanding, this approach goes above and beyond in doing as such. This strategy requires putting away catchphrases and the things they are related with in an outside information source. By doing this, the watchwords become autonomous of the automation instrument being utilized to run the tests. A rundown of catchphrases is one of the components of a content that is utilized to test programming. These terms mean different activities. These watchwords may likewise be alluded to as "clicklink" or "verifylink," and the items might be "submit buttons" or "login usernames." The names might be pretty much as clear as "click" or "login" or as convoluted as "clicklink" or "verifylink." For this procedure to function as planned, a common item vault that can plan objects to the exercises related with them is important. **[13].**

6) Hybrid Testing System, number six

The crossover system is a blend of at least two unmistakable structures that was made to take the best components from every structure that addresses the issues of automation. This is accomplished by minimising the downsides of some frameworks while maximising the advantages of others. Because every piece of software is unique, the procedures that are used to test it should also be distinct. It is essential to have a flexible framework for automated testing as the number of teams using an agile methodology increases. When it comes to getting the greatest possible test results, a hybrid framework is easier to modify **[13].**

AUTOMATED TESTING'S INTRODUCTION AND EVOLUTION

Software development is a time-consuming and lengthy process. The process consists of several stages, including Evaluation, Requirements, Analysis, Design, Implementation, Validation, and Deployment. Validation testing is typically conducted during the later stages of software development, as compared to other stages in the process. This is a time when the pressure of software delivery is higher than that of other tasks. Automating the testing process will help the project go more quickly, among other things**. [16]** The testing phase is estimated to require up to 50% of the project development resources. It is often claimed that this process is costly. Furthermore, there are several factors that can contribute to the missed deadlines for delivering a successful software product. These factors include last-minute changes in requirements and issues related to developers, such as taking leave or changing jobs. In certain situations, developers may encounter difficulties that could cause them to miss deadlines. This could lead to delays in completing the project and potentially result in a more rushed and costly testing process. Additionally, organizations may have a preference for thoroughly testing software systems while minimizing the time and resources required. As a result, numerous organizations are increasingly recognizing the value of automated testing and are adopting it as a preferred approach. **[17]** Automated testing involves the use of automated tools to test software programs. Automated testing is facilitated by computer-aided software engineering, in simpler terms. When manual testing is performed, the responsible person or team has the ability to plan, design, and execute the tests. **[18]** Software test automation involves the utilization of software, rather than individuals or teams, to manage the execution of tests, compare actual outcomes with expected outcomes, establish test preconditions, and perform other functions related to test control and reporting. However, developing tools for automated testing is a costly and time-consuming task **[19]**.

The automated tool is characterized as software that checks for the presence of certain program attributes such as the appropriateness of the program module interface, the correctness of the program control structure, and the correctness of the program syntax. **[20]** An automated test framework is a collection of abstract ideas, techniques, and working conditions that will be used to design, develop, and perform automated tests. This definition of an automated test framework is intended to be a broad one. Additionally, it incorporates the physical structures that are employed for the formulation and implementation of tests, in addition to the logical interactions that take place between the various components of the test.

Automated testing is advocated in a variety of contexts, including the following:

- When compared to manual testing, automated testing can be carried out more quickly and effectively.
- The automated testing can handle even the most complicated of systems. Before putting the software systems through their ultimate deployment, the testing must first be completed.

- If there are a large number of opportunities for regression testing, also known as repeated testing, then automated testing is an easy way to support these repetitions. Not only will it be useful for carrying out the testing, but it will also be useful for producing comparison log files.
- If the size of the software is large and manually testing it poses a significant challenge, then it is recommended that automated testing be carried out instead.
- Due to the extensive testing resources required, many black box tests, such as stress tests, load tests, and durability tests, can only be carried out automatically. Only simulated testing offers a viable alternative for these kinds of examinations. **[21]**

Use of automated testing is not advised in the following specific situations:

- The expensive cost of implementing automated testing is one drawback.
- The creation of tools for automated testing is incredibly complex and time-consuming, and it calls for a significant investment of resources. Because of this, automated testing is only recommended in circumstances like these where there is an actual requirement for it.

The benefits of automating testing

- By enabling practically unattended test case running, automation testing makes test case execution simpler. At the conclusion of the procedure, this offers the chance to check the outcomes. The effectiveness of the application is improved by streamlining the overall test execution.
- Automation testing improves the reliability of tests by ensuring equal focus on all areas of testing, resulting in a high-quality end product.
- Increasing the amount of test coverage can be achieved by utilizing automation testing. With this method, a bigger number of test cases may be created and run for the application under test. Increased test coverage and the capacity to identify more bugs arise from this. Because of this, it is possible to test applications with more complicated architectures and with more features.
- Minimizing Human Interaction: Automation testing eliminates the possibility of human error caused by negligence, as every aspect, from test case creation to execution, is fully automated. This lessens the need to fix bugs once the product has been released.
- Time and money are saved with automation testing. Even though the initial investment could be greater, it ends up saving money and time in the long term. The quick formulation and execution of test cases is credited with the excellent caliber of the work.
- As automation testing efficiently records any problems that occur, it enables earlier defect discovery. This documentation makes it simpler for the development team to find and fix errors, which leads to a quicker turnaround. Early fault detection makes fixing it simpler and more affordable.

Disadvantages associated with automation testing

- The high initial cost is one of the downsides. Early project implementation of automation testing requires a large time and financial investment. A lot of work needs to go into choosing the right tool and creating unique software.
- It is not possible to achieve 100% test automation. In general, the goal is to automate as many test cases as possible. However, in practical real-world scenarios, it is not always feasible to automate every test case. For close observation, some test scenarios may call for human interaction. Testing usually includes a human aspect because it can't assess everything that people can, such design and usability.
- It is not possible to automate all types of testing. Automating tests to verify the user-friendliness of the system is not feasible. In a similar vein, testing for graphics or sound files cannot be simply automated. Because textual descriptions are frequently used in automated tests to check the output.

- A prerequisite for this position is programming knowledge. Every automation testing tool utilizes a specific programming language for writing test scripts. For automation testing, programming skills is necessary.
- False positives and false negatives: Occasionally, automation tests will fail, giving the appearance that the system is broken. However, these failures may not actually indicate any issues. In certain situations, the tests may produce false negatives if their purpose is to confirm the existence of functionality rather than ensuring that it functions correctly.

Applications for Software Testing Automation

The fact that not everything in a project can be automated must be emphasized. Automated tests are not intended to replace manual tests, but rather to work alongside them and complement each other. The manual system needs to be tested to ensure that it is understandable. This involves understanding the steps of the functionalities so that they can be recreated in an automated script, or programming code, later on. Software testers must constantly hone their current abilities and pick up new ones in order to match the changing demands of their job. Additionally, it's critical to adhere to a method in order to accomplish it correctly [22]. The first stage entails selecting the appropriate automation tools. The scope of the application where automation will be used must then be specified. The planning, design, and development phases are prepared first before executing the test scripts. In order to begin seeing positive results, it is essential to consistently maintain the software's functionalities. According to **[23]**, it is recommended to automate certain practices and processes. These comprise test cases that demand several iterations, test cases that concentrate on application features that provide a high risk, test cases that must be conducted with various data sets, and test cases that must deal with enormous amounts of data.

→ The essential features for the company are referred to as the critical pathways of the system.
→ These functionalities have a high mistake rate and are difficult to monitor during manual testing.
→ Tests that must be run on various hardware, software, and web browsers.

There are some things, nevertheless, that should not be mechanized. These consist of unreliable features that are under development or could change. Additionally, unpredictable results, such as Captcha, cannot be effectively evaluated by automated tools since the specific image shown in the user interface is unknown. The planning, design, and implementation of automated testing may involve more time than may be saved by the tool in the short term, so projects with limited time and resources may not be suited for automation. Please provide more contexts or specify what kind of tools you are referring to. In order to ensure the successful implementation of Testing Automation, it is crucial to have appropriate tools that can effectively guide the automation process. By utilizing these tools correctly, organizations can reap significant benefits **[24]**. These tools are available to all individuals and organizations, either through open-source or licensing options. The fact that not all systems are automated in the same way must be noted. For the automation test to be effective, it is essential to carefully choose the tools that are in line with the unique project needs. This entails selecting tools that work with the technology that was used to create the application being tested. The current key functions of STA are constructed using the approaches of Agile, DevOps, and Continuous Integration. These functionalities should also be in line with the latest technological trends, as mentioned by **[25]** and **[24]**.

The main goal is to maximize these trends' positive effects, which are mostly related to AI and machine learning. The following factors must be taken into consideration when selecting STA tools:

→ Support for API and services testing

→ User-friendly interface

→ Compatibility with the project's scripting language

→ Analytical capabilities and capabilities in artificial intelligence/machine learning (AI/ML) Our platform offers support for multiple frameworks and more.

Each of these features undergoes distinct evolution over time, but their ultimate goal remains consistent: to deliver high-quality results quickly. [24] asserts that a good tool should be able to enable fundamental optimization, automate test case and data collection, offer analytics, and deliver better solutions. Selenium, Test Complete, Katalon Studio, and QTP are some of the best automated testing solutions. However, there are a tonne of additional choices available.

QUALITY ASSURANCE AUTOMATED TESTING TOOL

Software testing is crucial to the software development life cycle and to ensuring the calibre of the final output. The tests' execution can then be managed by the proper software, and the outcomes can be contrasted with those that were predicted. Test automation is the term for this procedure.

The automation test tool creates an environment where computerized tests can be run. The norm for automation for a particular type of product is established by an inherent structure. An effective tool is a combination of many diverse strategies, programming standards, viewpoints, techniques, conventions, system hierarchies, modularity, coverage mechanisms, and test data injections. These sections function as distinct parts that must be put together into a whole in order to properly explain a business procedure. The customer can take advantage of a number of capabilities that this tool provides to build, execute, and testify on automation test scripts quickly and effectively. For software products with a longer anticipated lifespan, these automated tests are a wise investment because they can run quickly and on a frequent basis. This method aids in the organization of test suites, which in turn aids in the enhancement of testing process effectiveness. A structured testing tool can make it simpler to get rid of automated test case duplication within an application. [26]

A testing tool can be utilised with any application, regardless of the issues that the application that is being tested is having (such as components, stacks, structural designs, etc.). A testing tool is always independent of the application that is being tested. Newly created test cases are constantly added to the existing automation in order to stay up with the development of the programmer under development. The fact that test automation cannot completely cover all tests must be understood. The importance of remembering this cannot be overstated. It is important to consider both the cost and the quantity of work required when deciding which types of tests should be automated first. Automating test cases that cost a lot but involve little work should happen as soon as possible. The next step is to further automate test cases that have a history of errors, are regularly modified, and are utilised alongside test cases that require little to no work. Test automation, which also makes it possible to execute tests without the involvement of the end user and guarantees correctness and repeatability, might help testers feel less aggravated. Instead, at this point, testers can focus more on sophisticated test scenarios. [27] Defects with high testability can be found via randomly generated tests. We are able to conduct a range of tests quickly and effectively with the aid of this testing automation solution. For the following, this testing device is responsible:

→ Establishing a way to connect to or operate the application that is being tested.
→ Providing a framework for how expectations should be expressed.

→ Carry out the various tests.

→ Put the findings on the record.

Test Strategy

An illustration of the testing approach used during the software development life cycle is called a test strategy. By outlining a precise plan for achieving the test objectives, this document eliminates all ambiguities and unclear requirement descriptions. The testing purpose, testing kind, total time, needed resources, features and functions to be tested, risk analysis, defect resolution, process improvement, and test environment are all included. This testing plan provides the tool required to put our testing methodology into practise. Based on the system's particular application, a unique testing approach should be developed for each system. The process known as a test strategy outlines how to effectively accomplish the testing objectives using the resources at hand. Thus, for project stakeholders, the test strategy clarifies the testing methodology. [28]

The following should be covered by a thorough test strategy:

1) Scope of

Outline the goal of the application and talk about testing. It contains details like who will review and recommend this document. Define the testing procedures and phases in accordance with the project's schedule as a whole.

2) Test Purposes

The test application should be assessed according to how closely it adheres to the requirements and customer acceptance standards. As a result, these requirements and acceptance criteria must be applied generally to particular test plans that confirm and evaluate the anticipated results for each test to be carried out [29]. The goals must be divided into groups based on importance and risk. It is necessary to test features and functions. With the exclusions shown, all features and functionalities must be scheduled for testing enclosure or omission. Due to a lack of hardware or control, some functions might not be able to be tested [30]. The inventory should be organized according to functional area for clarity.

3) Testing Method

This methodology defines the essential details for describing the stages and types of testing. Include details such as when testing should begin, the test owner, the duties to be performed, the testing approach, and, if applicable, the automation strategy and tool.

4) Test Conditions

The list of environments and the necessary configuration for each environment must be included in the setup of test environments. The test environment is made up of elements that configure the software, hardware, and network to facilitate test execution. To detect any environment- or configuration-related errors, the test environment's configuration must be identical to that of the production environment.

5) Process Enhancement

Process improvement must be the main focus of the testing process throughout. This testing strategy must specify how progress will be tracked and regular feedback will be given. This viewpoint may improve the testing process, products, and metrics.

6) Successful outcomes

Deliverables should be listed in their entirety, along with their locations. The deliverables include test cases, test data, a test summary report, and a software quality assurance report.

7) Schedule

A single master testing schedule should contain all testing operations. There must be estimated times for each job in this plan. To provide oversight of all processes, testing resources must be given to each task and quality gates must be set.

COMPARISION OF MANUAL VS AUTOMATED QA PROCESSES

⇒ **Software Testing Techniques**

A. <u>Manual Testing</u>

The most basic and exhaustive type of software testing is manual testing. The most important test-related step is this one. Rather than employing any specialised software, these software testers create test cases by hand. There will always be some manual testing. The application must first be manually tested before we can determine whether automation is feasible. The use of testing tool knowledge is not necessary. Software testers develop test suites on an Excel sheet using their own knowledge and logic in this method. **[31]**

1) Goals

⇒ The objectives of manual testing are to find flaws and problems in a software application, confirm that it complies with the requirements, and make sure it works well and satisfies the demands of the end users.

⇒ Manual testing's main objective is to make sure the software is error-free. The system should be created to meet the precise functional specifications established by the clients.

⇒ The proportion of test coverage for test cases created during the testing phase is high. The developers have agreed to test any disclosed flaws. Manual retesting should be done by testers on the corrected flaws.

2) Negative aspects of manual testing

⇒ Software testers may have a busy schedule as a result of manual testing. Furthermore, it takes a lot of time. It can get very boring to run the same test cases repeatedly for the same release. Automation becomes a useful tool for the consumer in these circumstances.

⇒ When software is manually tested, there is a greater chance of errors.

⇒ Manual testing requires testers to dedicate a significant amount of time and attention.

3) Test Case

To ensure that a certain module or code is functional, a test is carried out. What will be tested is described in the text. Additionally, it describes the kinds of data that should be used as input and lists the steps taken to confirm the outcome. The actual result is examined to see if it conforms to the predicted outcome. Typically, a test case has three parts: an expected response, an action or event, and an input. A test case is used to check whether a feature of an application is operating in line with the precise specifications established by the client. Writing test cases is mostly done to verify the application's test coverage.

4) Structure of test cases

A crucial factor to take into account is the structure of test cases.

Included are the following parameters:

1) Test Scenario ID first, followed by Test Scenario Description.
2) Test Case Id,
3) Test Case Description.
4) Test Steps No. 5
5) Prerequisites
6) Test results
7) Post-conditions
8) Expected Outcome
9) Actual Outcome
10) Status
11) Test case conducted
12) Test case executed
13) Test case executed
14) Comments.

- Test Scenario Id: This unique identification number is used to reproduce bugs or scenarios.
- Test Scenario Description: This section outlines the procedures for reproducing the bug or circumstance.
- Test Case Id: This is a specific identification number for a test case.
- Test Case Description: This is a description of a test case.
- Test Steps: These are the steps taken to execute test cases.
- Preconditions: The first conditions needed to create test cases.
- Data used as testing input is known as "test data."
- Post conditions: The conclusion that must be true following the remaining test phases.
- Expected Outcome: This is the functionality required for the programmer to operate as intended.
- Actual Result: During testing, this is what we observe in the application.
- Condition: It shows the test cases' present condition.
- It is the tester's name who carried out the test case.
- Test case executed date: The day the tester actually runs the tests.
- Remarks: It contains remarks on the development of test cases.

5) Test Case Execution Process

In software testing, the execution of test cases and their outcomes are crucial. Improved product quality results. Every single activity is backed up by documentation. The following duties must be taken into account by the software tester when running test cases.

⇒ Record of test case execution success and failure in a word document.
⇒ Record of test case execution duration for the test suite.
⇒ Total number of test cases performed.
⇒ Amount of flaws found during testing.

B. <u>Automated Testing</u>

With the help of this method, test cases can be created and run automatically. Automated testing is the name given to this technique. Using the proper software, the tester writes application-testing programmes during this stage. It simply involves the automation of a manual procedure so that no human involvement is required. The manual testing process is made easier by automated testing. **[32]**

1) Goals

⇒ Automated testing's goal is to boost software quality and testing effectiveness. Automation seeks to minimize the number of test cases that require manual execution.
⇒ The test data is entered into the system by the automation software, which then contrasts the actual results with the expected ones. It produces complete test reports as a result.

2) Advantages of Automated Tests

- Comprehensive testing of application functionality.
- Automated test case execution is 70% faster than manual test case execution..
- Time and financial savings.
- Enhances dependability.
- Guarantees consistency.
- Delivers more precise results.
- No utilization of human resources during test case execution.
- Increased test execution speed.
- Automation makes the process much more interesting as opposed to manual testing,
- Which eventually becomes boring?
- Possibilities for recording test suites and playing them back as needed.

3) Automated Testing Process

There are five stages to the automated testing process:

I Test tool selection: The technology and application play a large role in the automation tool selection.
II II Define the scope of automation the portion of the application being tested that will be automated is known as the automation scope.
III III Development, planning, and design The following information is created as part of the planning and strategy for automation during this phase:

⇒ Automation tools have been chosen.
⇒ The framework's design and functionalities.
⇒ Automation elements that are both in and out of scope.
⇒ Automation testbed setup.
⇒ The scripting and execution schedule and timeline.
⇒ Automated testing outputs.

IV Execution of the test: Automation scripts are run at this stage. Before test cases can be run, the script requires input test data. They offer complete test findings once they have been executed.

V Maintenance: Automation scripts must be added, reviewed, and maintained for each release cycle because the system under test is expanding with each cycle. To increase the productivity of automation scripts, upkeep is now necessary.

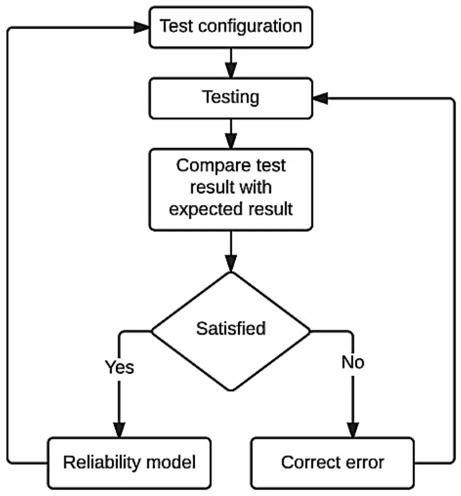

Figure 1: Automated Testing Model

Table 1: Manual vs. Automated Evaluation

Parameters	**Manual Testing**	**Automated Testing**
Reliability	Manual testing is not accurate at all times due to human error; thus it is less reliable.	Since it is performed by third-party tools and/or scripts, therefore it is more reliable.
Investment	Heavy investment in human resources.	Investment in tools rather than human resources.
Time efficiency	Manual testing is time-consuming due to human intervention where test cases are generated manually.	Automation testing is time-saving as due to the use of the tools the execution is faster in comparison to manual testing.
Programming knowledge	There is no need to have programming knowledge to write the test cases.	It is important to have programming knowledge to write test cases.
Regression testing	There is a possibility that the test cases executed the first time will not be able to catch the regression bugs due to the frequently changing requirements.	When there are changes in the code, regression testing is done to catch the bugs due to changes in the cod

TRANSITION FROM MANUALLY OPERATED TO AUTOMATION TESTING

Over the course of recent history, the market has been increasingly reliant on advancing technological capabilities, as businesses have been pressured to speed up their delivery times in order to remain competitive. Because of the rapid development and application of agile approaches, the adoption of automation technologies has become obligatory and must now take place.

Manual testing at some companies produced satisfactory results, but it was ineffective in certain testing domains that needed multiple iterations of the test code. These domains failed the test. The manual testing procedure, when applied in an environment with increased scale, consumed a significant amount of testing time and resulted in unwelcome delays. **[33]**

The development of software that automates testing opened the door to the potential of doing iterative and parallel tests on a number of different kinds of devices, browsers, and operating systems. It did it automatically, producing full reports complete with error logs. The dynamics of the industry were eventually altered as a result of these traits. Automation testing became the ideal alternative for trustworthy software testing services as a result of its maximum device coverage, quicker execution time, and low testing costs.

- **Testing software with automated systems is more reliable than testing with traditional approaches.**

A manual tester runs the risk of forgetting some specific tests. It is imperative that whatever a developer develops and contributes to the test automation suite is not overlooked. Additionally, the human tester may decide to skip certain tests in the mistaken belief that the feature in question has already been tested. Because automated testing eliminates the need to repeat tests, test repetition is not required.

- **Automation increases the amount of tests performed.**

Automated software testing has the potential to increase the scope of your test while also enhancing the product's quality. Automation now allows for the performance of lengthy and laborious manual testing, which were previously avoided. In addition to this, it enables you to conduct tests on several systems, each of which might have its own unique configuration. Software test automation makes it feasible to conduct thousands of complicated tests simultaneously and in a seamless manner, providing test coverage in a manner that is not achievable with human testing.

- **Quality of the Tests Has Improved**

The software testing process can be automated such that the same processes are carried out over and over again without any difference. In a similar vein, there are no mistakes in the test results that were created by the automated tests. Because of this, the quality of the tests has increased, and manual testers are now able to devote more time to the production of new tests and concentrate more on complicated issues rather than monitoring the same repetitive task.

TESTING FOR INDUSTRIAL CONTROL SOFTWARE MANUAL & AUTOMATED

While making programming for modern control, testing is a critical stage in the designing system. To help programming engineers in the testing of their items, specialists have proposed various techniques for making great experiments. In the new years, an extensive variety of automated test producing strategies [34] have been explored determined to improve manual testing. Regardless of proof to recommend that consequently produced test suites might try and cover more code than those manually created by engineers [35], this doesn't suggest that these tests are valuable as far as recognizing issues. The benefits and detriments of each approach ought to be differentiated and thought about additional totally in similar examinations since human testing and automated code inclusion coordinated test creation are essentially unique and every method has its own inborn restrictions. In this examination, we direct an observational assessment of automated test age and differentiation it with test suites manually made by modern specialists on 61 projects from a genuinely modern train control framework. IEC 61131-3 [36] is a well-known programming language utilized in the making of control programming in the security basic industry [37]. This framework utilizes programming made in that language. We have made a state of the art test producing instrument for IEC 61131-3 programming called COMPLETETEST [38] and looked at the code inclusion, cost, and shortcoming location of it to manual testing done by modern designers. As per the discoveries of our contextual analysis, automated test age can create tests with code inclusion that is equivalent to modern designers' manual testing in a little part of the time. IEC 61131-3 guaranteed wellbeing basic control programming is in concern. Automated test creation can possibly save testing time by up to 90% while planning programming as per IEC 61131-3. In any event, when complete code inclusion is achieved, test suites worked by PCs are not generally better compared to test suites made by people at recognizing issues. 56% of the test suites worked with COMPLETETEST found less defects than the test suites manually created by modern architects. This was found through our contextual analysis. By and large, apparently human made tests can distinguish more unambiguous types of defects (such legitimate substitution, refutation inclusion, and clock substitution) than PC produced tests can. On the off chance that, notwithstanding primary elements, the automated test producing strategy involved the recognizable proof of these particular adjustments as the inclusion necessity, we could make test suites that were more compelling.

IEC 61131-3 [37] has become a very popular programming language that is used in an extensive variety of control frameworks, from traffic the board programming to control frameworks in thermal energy stations. In the motivation behind making modern control programming, this language is utilized. For IEC 61131-3 programming, a few automated test creating strategies have been advanced throughout the course of recent years, including [39] and [40]. It has been shown that these methodologies produce high code inclusion for an assortment of IEC 61131-3 modern programming ventures, and they can frequently foster test suites for a specific code inclusion basis. Before, extraordinary code inclusion has been utilized as a substitute for a test suite's capacity to track down shortcomings. Be that as it may, as per ongoing exploration by Inozemtseva et al. [41], inclusion ought not be used as a quality marker since it is a temperamental indicator of shortcoming discovery. At the end of the day, the viability of a test suite to distinguish issues might rely more upon different components of the test plan than it does on how intently the code's construction is reproduced. The use of both manual testing and automated inclusion based test age for genuine software engineers is being considered. Numerous scholastics have centered their contextual analyses [42], [43], and [44] on manually developed test suites that were at that point being used, while others have directed controlled tests utilizing human members who made both manually and consequently produced test suites. These outcomes persuaded us to explore the overall benefits of manual testing versus automated inclusion based test advancement with regards to modern wellbeing basic control programming. For testing that depends on both the particular and the execution, unique prerequisites should be met for these sorts of frameworks.

USE OF AUTOMATED TESTING TOOLS TO ENSURE THE QUALITY OF MOBILE APPLICATIONS

Programming testing empowers programming analyzers to find shortcomings in the product and fix them, which at last further develops the product's general quality. In the field of programming advancement, where it is by and large considered fundamental, testing programming has recently become standard method. Both human and automated programming testing techniques are accessible. While testing is done manually, experiments are composed the hard way and run without the guide of any automated testing programming. In manual testing, an analyzer runs the tests via cautiously exploring through the framework's numerous connection points, testing with different info amounts, noticing the consequences of the tests, and contrasting those outcomes with the tests' normal results. Automated testing must be finished with the guide of an automated testing apparatus. Rather than manual testing, the automated testing instrument conveys testing that is overseen by a PC. The testing device executes the experiments to evaluate the usefulness and execution of the item under audit. While reducing the amount of human labour required for manual testing is the aim of automated testing, this strategy in no way does away with the need for manual testing [45]. Because there are now so many software options accessible to users for use on handheld and portable devices, mobile platforms are becoming more and more popular all over the world.

Mobile application development is also using testing as a way of quality assurance [46]. A wide range of various tools have been proposed and used to achieve this objective. In terms of the unique features they provide, the platforms they are compatible with, the amount of code they cover, and how effectively they operate, these tools have already been evaluated and contrasted. On the other hand, the many quality aspects that the various automated testing approaches for mobile applications might enhance in tested apps have not been compared or evaluated. Due to this, two research objectives for this study have been defined, and they are as follows:

⇒ To survey different portable application testing approaches with an emphasis on which quality perspectives they help the applications under test accomplish;
⇒ To follow the general patterns of vital quality variables accomplished in the portable applications under test utilizing automated testing procedures.

The objective of this study was to improve or enhance specific valuable quality components already existing in the tested mobile applications by reviewing and comparing various automated testing approaches. The study's conclusions and outcomes are useful for both scholars and practitioners. This knowledge will help both parties become more effective in their work. Each app has a different set of quality requirements that must be met. In order to assess diverse mobile applications, it is required to select a number of testing tools. The practitioners may therefore discover that they need to look for tools that will help them ensure the desired quality characteristics in an application that is currently being tested (AUT). The study's quality characteristics may need to be taken into account by researchers who are interested in presenting techniques and frameworks for grading mobile apps. Additionally, students are able to start their own research study using these tools in order to propose solutions that are combined, revised, and enhanced in order to reach the highest number of high-quality traits in the AUT.

How fruitful a product item will be generally rely upon the quality of the program. This sets out a sizable freedom for programming quality assurance inside the product business, which is inspired by client satisfaction. It is presently fundamental to foster an item that is of brilliant quality and without any trace of any deformities while remaining inside the requirements of the accessible time and cash. Placing such items into utilization with practically no difficulty or even any is an overwhelming test. This is the defense for how the idea of programming testing was created [47]. The product testing period of the product improvement life cycle (SDLC) has created inside the product business to turn into an exhaustive and vital stage. It likewise offers an extensive investigation of different systems, including necessity elaboration, programming plan, and coding [48].

To guarantee that product addresses client issues and is of the ideal type, testing is a cycle that is utilized to evaluate the rightness and usefulness of the product [49]. As per the IEEE, programming testing is the most common way of evaluating a framework or its parts to decide if it fulfills client necessities or to find out how much the genuine outcome digresses from the expected result [50]. As a result, the goal of software testing is to run a programme to find flaws and features that are missing from the product but were expected by the user requirements. The quality and effectiveness of the software system will increase as a result of properly executed software testing.

The expense of programming upkeep can be diminished by recognizing issues in the product and adjusting them before discharge. Each of the exercises remembered for the product testing interaction can be finished either manually or consequently. The most well-known sort of programming testing is done manually. Manual route is performed to finish the job inside the product's application. A test plan or experiments are many times followed while performing manual testing. The experiments offer an exhaustive defense of the testing climate as far as the errands that should be finished during testing. Contrarily, when performing automated testing, the testing is done through the use of a specific testing tool rather than by manually moving through the various components of the application under test. At first, manual testing was the sole sort of testing carried out. Because human error can occur during manual testing, it's possible that some defects will go undiscovered or undetected. Therefore, manual testing cannot ensure an improvement in the quality of a software system. This flaw in manual testing has given rise to the discipline of automation testing. With the aid of automated testing, the testing procedure can be accelerated. Significantly more accentuation has as of late been paid to computerize testing, and numerous analyzers progressively favor involving it for an assortment of programming frameworks [51]. The basic component of the automated testing procedure is the automated testing instrument that is utilized to do the tests.

CONCLUSION

Finally, this comparison of human and automated Quality Assurance (QA) testing techniques demonstrates the significant impact of automation in the field of software quality assurance. The study's findings highlight how automation, with its efficiency, repeatability, and ability to perform repetitive tasks, has become a cornerstone of current QA methods. It shortens testing cycles, reduces human errors, and contributes to faster software releases, in line with agile and DevOps techniques. Nonetheless, it is critical to acknowledge that manual testing is still necessary in some situations, notably those needing subjective judgment or exploratory testing. The human touch cannot be completely replaced in quality assurance, and QA professionals must adapt their skill sets to focus on more strategic and creative parts of testing, such as test case creation, risk assessment, and customer-centric testing. Furthermore, enterprises must carefully balance the advantages and disadvantages of automation, taking into account aspects such as initial setup costs, ongoing maintenance, and the nature of their software projects. The most practical strategy appears to be a hybrid approach that combines manual expertise with automated test suites. In conclusion, automation has a revolutionary impact on quality assurance testing, streamlining processes and increasing productivity, but it should be considered as an additional tool rather than a complete replacement for human testing expertise. The success of QA efforts in the digital age is dependent on a balanced integration of both manual and automated testing approaches to maintain software quality and fulfill the industry's ever-changing needs.

REFERENCES

1. Yinnan Z, Xiaochi W. Implementation of Software Testing Course Based on CDIO. IEEE International Conference on Computer Science & Education. (ISSN 978-1-4244-9718-8). 2011; 11:107-110.
2. Nicolò Paternoster, Carmine Giardino, Michael Unterkalmsteiner, Tony Gorschek, Pekka Abrahamsson, Software development in startup companies: A systematic mapping study, Information and Software Technology, Volume 56, Issue 10, 2014,Pages 1200-1218, ISSN 0950-5849, https://doi.org/10.1016/j.infsof.2014.04.014.
3. Singh Y. Software Testing. (1st Ed.). INDIA: Cambridge University Press, 2012.
4. Myers, G. J., Sandler, C., & Badgett, T. The art of software testing. Hoboken, NJ: John Wiley & Sons, 2012.

5. Barus, A. C., Hutasoit, D. I. P., Siringoringo, J. H., & Siahaan, Y. A. White box testing tool prototype development. 2015 International Conference on Electrical Engineering and Informatics (ICEEI), 2015, pp.417–422. Doi: 10.1109/iceei.2015.7352537
6. Mahajan, P., Shedge, H., & Patkar, U. Automation Testing In Software Organization. International Journal of Computer Applications Technology and Research, 4 (vol. 5), 2016, pp.198–201. Doi: 10.7753/ijcatr0504.1004
7. Dike, S. The Pragmatic Programmer: A Book Review. 22 May 2018, Retrieved May 21, 2020, from https://blog.thedigitalgroup.com/9-reasons-automation-testingis-key-to-successful-software-development
8. Fernandes, J. D., & Fonzo, A. D. When to Automate Your Testing (and When Not To), March 2013.
9. Ramler, R., & Wolfmaier, K. Economic perspectives in test automation. Proceedings of the 2006 International Workshop on Automation of Software Test - AST 06, 2006, pp.85-91. doi:10.1145/1138929.1138946
10. Abraham, C. Aspire Systems - Technology Solutions: Digital Transformation. 17 Sep 2019, Retrieved May 22, 2020, from https://www.aspiresys.com/
11. Raulamo-Jurvanen, P., Mäntylä, M., & Garousi, V. (2017). Choosing the Right Test Automation Tool. Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering - EASE17, 1–11. Doi: 10.1145/3084226.3084252
12. Oliinyk, B., & Oleksiuk, V. Automation in software testing, can we automate anything we want? 2nd Student Workshop on Computer Science & Software Engineering, 2019, pp. 224-234.
13. Aebersold Kirsten, 'Test Automation Frameworks'. [Online]. Available: https://smartbear.com/learn/automated-testing/testautomation-frameworks/. [Accessed: 26-Aug-2019].
14. A. Bhargava, Designing and implementing test automation frameworks with QTP : learn how to design and implement a test automation framework block by block. Packt Publishing, 2013.
15. 'A Guide to Automation Frameworks |Smartsheet'. [Online]. Available: https://www.smartsheet.com/test-automation-frameworkssoftware. [Accessed: 30-Aug-2019].
16. Ruparelia, N. B. (2010). Software development lifecycle models. ACM SIGSOFT Software Engineering Notes, 35(3), 8-13.
17. Amaricai, S., & Constantinescu, R. (2014). Designing a Software Test Automation Framework. Informatica Economica, 18(1).
18. Chang, E., & Dillon, T. S. (1997). Automated usability testing. In Human Computer Interaction INTERACT'97 (pp. 77-84). Springer, Boston, MA.
19. Dustin, E., Rashka, J., & Paul, J. (1999). Automated software testing: introduction, management, and performance. Addison-Wesley Professional.
20. Farid, M. R., & Abraham, K. (2010). Automated Database Applications Testing: Specification Representation for Automated Reasoning (Vol. 76). World Scientific
21. Galin, D. (2004). Software quality assurance: from theory to implementation. Pearson Education India
22. Guru99 (2019). Automation Testing Tutorial: What is Automated Testing? [Website]. Available via: https://www.guru99.com/automation-testing.html [Accessed 08 Oct. 2020]
23. Quality-Stream (2019). Introducción a la Automatización de Pruebas de Software. [YouTube Video]. Available via: https://www.youtube.com/watch?v=R_hh3jAqn8M&ab_channel=Quality-Stream [Accessed 08 Oct. 2020]
24. Anderson, B. (2017). Best Automation Testing Tools for 2020 (Top 15 reviews). [Online post]. Available via: https://medium.com/@briananderson2209/best-automation-testing-tools-for-2018- top-10-reviews-8a4a19f664d2 [Accessed 20 Oct. 2020]
25. Brown, A.J., (2018). Testing trends in 2019 and further. QATestLab Blog. [Online Article]. Available via: https://blog.qatestlab.com/2018/11/06/testing-trends-2019/ [Accessed 13 Oct. 2020]
26. William EP. Effective Methods for Software Testing. (3rd Ed.). India: Wiley India Edition, 2012.
27. Naresh C. Software Testing Principles and Practices. (2nd Ed.) India: Oxford University Press, 2017.
28. Sebastian Elbaum "Test Case Prioritization: A Family of Empirical Studies" IEE Transactions on Software Engineering, Vol. 28, No. 2, February 2002
29. Yeresime Suresh, Santanu Ku Rath "A Genetic Algorithm based Approach for Test Data Generation in Basis Path Testing" at International Conference on Soft Computing and Software Engineering 2013.
30. Steven P. Reiss "Towards Creating Test Cases Using Code Search" at IEEE International Conference on Software Maintenance and Evolution 2014
31. Chauhan.N.2010. Software Testing : Principles and Practices. Oxford University Press
32. Divya Kumar , K.K. Mishra.2016. The impacts of test automation on software's cost , quality and time to market. 7th International Conference on Communication , Computing and Virtualization
33. Sujatha P, Sankar GV, Rao AS, Satyanarayana T. The Role of Software Verification and Validation in Software Development Process, IETE Technical Review. 2001; 18(1):23-26
34. Alessandro Orso and Gregg Rothermel. Software testing: a research travelogue (2000–2014). In Proceedings of the on Future of Software Engineering. ACM, 2014.

35. Gordon Fraser, Matt Staats, Phil McMinn, Andrea Arcuri, and Frank Padberg. Does Automated Unit Test Generation Really Help Software Testers? A Controlled Empirical Study. In Transactions on Software Engineering and Methodology. ACM, 2014
36. IEC. International Standard on 61131-3 Programming Languages. In Programmable Controllers. IEC Library, 2014.
37. Karl-Heinz John and Michael Tiegelkamp. IEC 61131-3: Programming Industrial Automation Systems: Concepts and Programming Languages, Requirements for Programming Systems, Decision-Making Aids. Springer, 2010.
38. Eduard Enoiu, Adnan Cau ˇ sevi ˇ c, Thomas J Ostrand, Elaine J Weyuker, ´ Daniel Sundmark, and Paul Pettersson. Automated Test Generation using Model Checking: an Industrial Evaluation. In International Journal on Software Tools for Technology Transfer. Springer, 2014.
39. Hendrik Simon, Nico Friedrich, Sebastian Biallas, Stefan HauckStattelmann, Bastian Schlich, and Stefan Kowalewski. Automatic Test Case Generation for PLC Programs Using Coverage Metrics. In Emerging Technologies and Factory Automation. IEEE, 2015.
40. Kivanc Doganay, Markus Bohlin, and Ola Sellin. Search based Testing of Embedded Systems Implemented in IEC 61131-3: An Industrial Case Study. In International Conference on Software Testing, Verification and Validation Workshops. IEEE, 2013.
41. Laura Inozemtseva and Reid Holmes. Coverage is Not Strongly Correlated with Test Suite Effectiveness. In International Conference on Software Engineering. ACM, 2014.
42. Xiaoyin Wang, Lingming Zhang, and Philip Tanofsky. Experience Report: How is Dynamic Symbolic Execution Different from Manual Testing? A Study on KLEE. In International Symposium on Software Testing and Analysis. ACM, 2015.
43. Jeshua S Kracht, Jacob Z Petrovic, and Kristen R Walcott-Justice. Empirically Evaluating the Quality of Automatically Generated and Manually Written Test Suites. In International Conference on Quality Software. IEEE, 2014.
44. Sina Shamshiri, Rene Just, Jos ´ e Miguel Rojas, Gordon Fraser, Phil ´ McMinn, and Andrea Arcuri. Do Automatically Generated Unit Tests Find Real Faults? An Empirical Study of Effectiveness and Challenges. In International Conference on Automated Software Engineering. ACM, 2015.
45. P. Rathi and V. Mehra, “Analysis of Automation and Manual Testing Using Software Testing Tool,” 2015.
46. D. Amalfitano, et al., “MobiGUITAR: Automated model-based testing of mobile apps,” IEEE Software, vol. 32, pp. 53-59, 2015.
47. X. Wang and G. He, “The research of data-driven testing based on QTP,” in 2014 9th International Conference on Computer Science & Education, 2014.
48. M. Monier and M. M. El-mahdy, “Evaluation of automated web testing tools,” International Journal of Computer Applications Technology and Research, vol. 4, 2015.
49. K. M. Mustafa, et al., “Classification of software testing tools based on the software testing methods,” in Proceedings of the International Conference on Computer and Electrical Engineering (ICCEE‟09), 2009, pp. 229-233.
50. G. Saini and K. Rai, "Software Testing Techniques for Test Cases Generation," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3, 2013.
51. N. Islam, "A Comparative Study of Automated Software Testing Tools," 2016. [8] D. Shikha and K. Bahl, “Software Testing Tools & Techniques for Web Applications.”