Journal of Theoretical and Applied Information Technology
31st August 2022. Vol.100. No 16
© 2022 Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

AGILE TEST AUTOMATION FOR WEB APPLICATION USING TESTNG FRAMEWORK WITH RANDOM INTEGRATION ALGORITHM IN MACHINE LEARNING TO PREDICT ACCURACY AND RESPONSE TIME ON AUTOMATED TEST RESULTS

V.VAMSI KRISHNA[1], **G. GOPINATH**[2]

[1]Research Scholar, School of Computer Science, Engineering & Applications,
Bharathidasan University,
Tiruchirappalli, Tamil Nadu, India.
[2]Professor, School of Computer Science, Engineering & Applications,
Bharathidasan University,
Tiruchirappalli, Tamil Nadu, India.

Correspondence Email: vamsikrishnavit@gmail.com

ABSTRACT

"Testing" must be a part of any software engineering approach that intends to build high-quality apps. By executing it with input values, testing seeks to find faults in the tested-object and develop confidence in its proper functioning. Web apps take first place in development and testing, according to everyday usage. By automating the entire software development testing process, testing automation saves time and money for developers and testers. Our proposed solution would use the "TestNG framework," an automated testing framework, to test a public website and save the test results in the format of a ".csv" or ".xls" file to a given directory. The Support-Vector-Machine Algorithm (SVM), Random-Forest Algorithm & other machine-learning algorithmshave been used to analyses the output file. The outcomes of all of the different ways will be compared and displayed on a graph. By Automating The Testing Framework Manual Testing Task Will Become Easy.

Keywords: *Test Automation, Web Applications, TestNGFramework, SVM, Random Forest Algorithm, Random Integration Algorithm.*

1. INTRODUCTION

The practice of building a programmed (test-script) in any programming/scripting language [1] that uses external automation helper-tools to duplicate manual-test-case methods. [2] stands for software-testing automation. It refers to the process of automating manual testing tasks. In order to test Software-Under-Test (SUT), software-test engineers should create & run a programmed [3]. To put it another way, toolkits are being created to test already-implemented source code. Its goal is to build test methods that can be automated [4]. Both writing test scripts and writing software are development activities; one is for the programmed itself, and the other is for the test scripts. Automating execution phase of software-testing cycle is quite familiar in automation business. “Software-Testing” is entirely automated by test script authoring is no longer a possibility; new toolkits are required to fully automate testing stages. [5].

Web-based apps are more dynamic and interesting than traditional programmed. As a result, traditional online app testing approaches and technologies are ineffective. A Web-application is an internet-based system that includes a database (or back-end) and interactive Webpages for users (or front-end)[6]. Web-applications are complicated software systems that are frequently updated and modified. Testing is both challenging and vital for them. It's tough because traditional testing methods and tools ignore the particular characteristics of web-based programmed, rendering them worthless. Testing web-based apps is dangerous because failure can be costly.

The agile approach is a software development project management style that involves self-organizing, cross-functional teams and their clients collaborating to create requests and solutions. Agile is a collection of ideas that emphasize adaptability and flexibility. In order to respond quickly to changing business needs, Agile places a premium on teams' ability to deliver in small, digestible chunks.

Based on the TestNG framework, this paper proposes a Web-application automation-testing approach. We'd put in place a system for automated Web-application testing. The format of '.csv' or '.xls' files was used in this proposed approach to automatically test the automation-testing framework. Each page encountered is analyzed using machine-learning algorithms to extract the necessary information. The data gathered will now be used to investigate the errors discovered while testing the web application.

2. RELATED WORK

Those who have contributed to this study include Z. Sun et al. [7] proposed a web-application testing platform based on hybrid automation software-testing framework that merge technical advantages of data & keyword driven software-test frameworks is proposed to address the issue of testing data differentiation & test case explosion in software-testing procedure of web-application system, as well as to lower the cost of automation testing maintenance.
Y. Kravchenko et al. [8] offer a dynamic Web-app verification technique primarily based totally at the shortest time criterion, in addition to a device evaluate. Dynamic verification techniques depend on the effects of real-global paintings as verified with the aid of using the software program machine or its prototypes to make sure that the effects are in compliance with the necessities and layout decisions. The consciousness of the evaluate could be on dynamic-verification-checking out and -tracking methodologies. The software program-verification aids jUnit (checking out) and AppDynamics (tracking) are assessed.
Reinforcement-Learning-Driven and Adaptive Testing (RAT) is defined by M. Amouei et al. [9] As an automatic black field checking out approach for coming across injection vulnerabilities in WAFs. SQL injection and cross-web website online scripting are of unique hobby to us due to the fact they`ve continuously been most of the pinnacle twenty vulnerabilities over the past decade. RATs, in unique, gather assault samples. It then employs a reinforcement mastering approach along with a ground-breaking adaptive seek set of rules to speedy locate definitely all circumventing attack patterns.
A. Sedaghatbaf et al. [10] describe ACTA, an automatic check introduction technique for black-container overall performance testing. ACTA does now no longer require a widespread amount of historic check information to examine the overall performance traits of the gadget below check due to the fact it's far primarily based totally on lively learning. Instead, uncertainty sampling is used to perceive which assessments ought to be finished at the fly. An ACTA permits for the specification of overall performance standards in phrases of circumstances, in addition to the improvement of assessments to fulfil the ones conditions. It is primarily based totally on an unsure sort of generative antagonistic networks.
S. Karlsson et al. [11] provided a technique for mechanically growing GraphQL queries inorder to check GraphQL API's. This has been done with the aid of employing a property-primarily based on totally approach to create question generator which is primarily based on total GraphQL illustration gadget beneath neath test. We are checking out a real-international software program gadget which suggests that this output is powerful & green at detecting actual flaws, overlaying a whole schema in seconds.
For evaluating app UIs, S. Yu et al. [12] advocate a widget feature matching & layout characterization matching-primarily based totally image-pushed mobile-app checking out device. We utilize pc vision (CV) technology which is used to carry out UI function contrast and format hierarchy extraction on mobile-application screenshots to reap UI systems that include wealthy contextual facts approximately app widgets which includes positions, relative relationship, and so on. Based at the UI systems we`ve uncovered, we are able to create a platform-unbiased take a look at script and function the goal widgets beneath neath take a look at. As a result, the proposed device employs a singular platform-unbiased take a look at-script encoding to non-intrusively replay take a look at scripts.
To achieve adaptive exploration of web-applications, Y. Zheng et al [13] gift Web

Explore, an automated give up-to-give up web-checking out framework. Web Explore makes use of curiosity-pushed reinforcement mastering to construct extremely good motion sequences (check cases) with sequential logical relationships. In addition, for the duration of the online-checking out process, Web Explore generates an automaton that serves as high-stage steerage to sell and enhance checking out efficiency. Six real-global projects, a industrial SaaS web-app, and a survey of the global`s pinnacle 500 web-apps had been all evaluated in depth.

A.Martin-Lopez.et.al. shows how specify & automate dependencies of inter parameter in web-application API's in their study [14]. We'll start with Inter-parameter Dependency Language (IDL), which is a domain specific language specifying relationship between input parameters in web-application services. To enable automatic IDL specification analysis, we offer a mapping that converts IDL documentation into Constraint-Satisfaction-Problem (CSP). We had proposed collection of 9 IDL document investigation methods that can be used to see if a given request, for example, meets all of the service's dependencies. Finally, for IDL specifications and analyses, we propose editor, parser, OAS extension, and constraint programming library.

T. Rangnau et al. [15] showsus to integrate 3 distinct automated dynamic software-testing approaches into CI/CD pipeline while still allowing for empirical measurement of the overhead introduced. Then we categorise and suggest preliminary answers to the DevSecOps community' various research/technology challenges. Our findings will aid us in making informed judgments as we deploy DevSecOps approaches in agile enterprise application engineering and corporate security.

H. Zhu et al. [16] provide a fascinating experiment for evaluating automation of oracle software. In common, software-testing is a process of inductive inference which tester tries to infer general qualities of a software system from its behavior on small number of test-cases. This debate looks at the theoretical foundations of software testing via the prism of computational machine-learning theories.

DARIO is a test oracle proposed by A. Arrieta et al. [17] that uses regression-learning methods to forecast the system's Quality-of-Service. The regression-learning algorithm in Oracle is trained on test data from earlier tested versions. Empirical estimation based on industry case study demonstrates that applicability of our technique. The regression tree method outperformed the other four regression learning algorithms that were tested. The regression tree technique's accuracy in predicting DARIO verdicts ranged from 79 to 87 percent.

Based on current approaches such as the keyword-driven and page-object models, B. Swathi et al. [18] present a methodology for constructing a test automation framework. The issue of test case generation and optimization has been addressed. Various encoding approaches regard test-case representation as a difficulty in soft computing techniques. It covers test case representation as well as test case generation using soft computing algorithms.

J. Kahles et al. [19] show how machine learning can be used in agile software testing to automate root cause investigation. After interviewing testing engineers, [20] It methodically extract essential information from raw log data (human experts). Despite finding only minimal correlations among specific clusters & failure underlying causes, remaining clusters' uncertainty necessitates labelling. Following new set of interviews with software-testing engineers, five ground-truth categories have been developed. It trained artificial neural networks [21] to undertake data categorization or pre-processing via clustering using manually labelled data [22].

3. EXISTING SYSTEM

"Manual-Testing" is a software testing framework in which a tester conducts all test cases by hand rather than utilizing automated technologies [23]. An approach for detecting bugs, flaws, and weaknesses in software is manual testing. Manual software-testing is most basic of all testing methods& it aids in detection of [24] major faults in software-based applications.

Many small businesses continue to rely on manual testing in the belief that it will provide them with greater accuracy [25]. Because they require a larger number of efficient resources to work on the automated testing process of their own software products, but they never anticipated it being a one-time event. We intend to address the existing issues in our proposed approach.

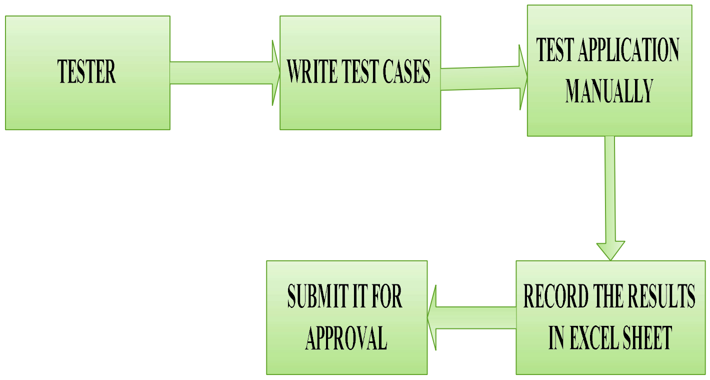


Fig:1 Diagram Of Existing System

4. PROPOSED SYSTEM

The term "Automation Testing" refers to a software-testing technique that compares actual result to expected result. This accomplished by using test scripts/any automated testing tool. "Automation-Testing" utilized to automate manual testing tasks that are time-consuming and difficult to complete. Here proposed work will test a public website using the "TestNG framework," an automated testing framework, and save the test results to a specified path in the form of a ".csv" or ".xls" file. Mixtures of machine-learning algorithms [26] were utilized to analyse the output file, including the Support-Vector-Machine (SVM) Algorithm, Random Forest Algorithm, and Random Integration Algorithm. All of the algorithms will be compared, and the results will be plotted in a graph.

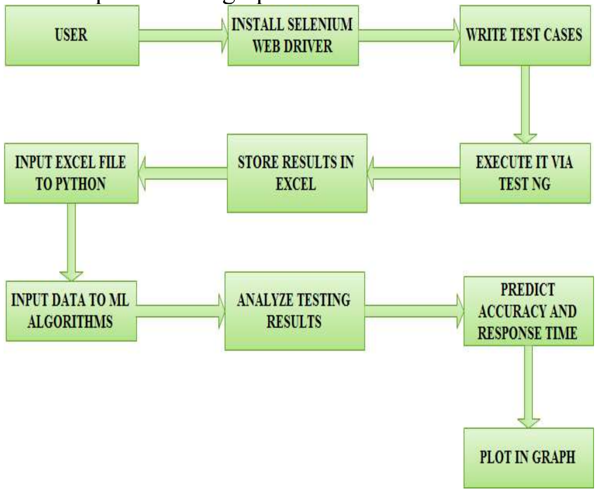

USER
INSTALL SELENIUM WEB DRIVER
WRITE TEST CASES
EXECUTE IT VIA TEST NG
STORE RESULTS IN EXCEL
INPUT EXCEL FILE TO PYTHON
INPUT DATA TO ML ALGORITHMS
ANALYZE TESTING RESULTS
PREDICT ACCURACY AND RESPONSE TIME
PLOT IN GRAPH

Fig 2 Block-Diagram Of Projectedsystem

5. METHODOLOGY

"TestNG"framework is used for software testing in the proposed Web-testing approach. The "TestNG" framework is a framework for automation testing, with the NG abbreviation standing for "Next-Generation." JUnit, which employs annotations (@), recommends TestNG. TestNG outperforms JUnit and is intended to make end-to-end testing easier. The steps of the projected approach, employs Machine-Learning algorithms, are depicted within Figure 2.

Pseudo code of SVM Algorithm

```
Commence
Initialised SVM parameter & structure
Produce an initial number of birthing lairs Bi = (q = 1, 2, 3, 4…n)
While (Stopping condition)
    If noise = false
        Look for proximity for a new lair by means of a Brownian walk Else Apply exploration for a way for a new layer by means of levy walk
    End if
    Calculate fitness of each new lair and evaluate with previous
    If
        B best, t > B best, p+1
        Select new lair
        B best = B best, p
    Else
        Go to 4
    End if
    Rank solutions;
    Return most excellent lair
    The overall greatest lair is fed to SVM classifier for training
    Training SVM classifier
End while
End
```

The distance of any line, ax + by + c = 0 from a given point say, (x0, y0) is given by d.

In the same way, the distance of a hyper-plane equation: $w^T\Phi(x) + b = 0$ from a given point vector $\Phi(x0)$ can be simply written as:

$$d_H(\phi(x_0)) = \frac{|w^T(\phi(x_0)) + b|}{||w||_2}$$

here ||w||2 is the Euclidean-norm for the length of w given by :

$$||w||_2 =: \sqrt{w_1^2 + w_2^2 + w_3^2 + \dots w_n^2}$$

Random Forest Algorithm

```
Input: X=Train data, R=Overall Features, r=members of features, T=Amount of tress
Output: Input data bagged class tag
Commence
1. While a particular tree in Forest T:
        i. Select a bootstrap sample M with size N out of training data.
        ii. Build tree Bgt through iteratively repeating the different steps regarding tree individual node.
            a. Choose randomly r from R
            b. Select optimal between f.
            c. Split node.
2. When the construction of T trees is executed, examples of Test data will be sent to individual tree & the allocation of class tag on the basis of majority votes will be carried out
End
```

For each and every decision tree, Random Forest computes nodes significance with Gini Importance, pretentious only 2 child nodes (binary tree):

$$ni_j = w_j C_j - w_{left(j)} C_{left(j)} - w_{right(j)} C_{right(j)}$$

Significance of each and every feature on decision tree is then deliberated as:

$$fi_i = \frac{\sum_{j:node\ j\ splits\ on\ feature\ i} ni_j}{\sum_{k \in all\ nodes} ni_k}$$

These can then be normalized to a value among '0' & '1' by dividing the sum of all feature significance values:

$$normfi_i = \frac{fi_i}{\sum_{j \in all\ features} fi_j}$$

Ultimate feature significance, at Random-Forest level, is its standard over all the trees. The sum of feature's importance value on each & every tree's is considered& divided by total number of trees:

$$RFfi_i = \frac{\sum_{j \in all\ trees} normfi_{ij}}{T}$$

Random Integration Algorithm

```
Input: X=Training data, R=Total Features, r=members of features, T=Amount of tress
Output: Input data bagged class tag
Start
1. While a distinct tree in Forest T:
    i.  Perform Tabu search algorithm to choose a best sample M with size N out of training data.
    ii. Build the tree Bt through iteratively repeating the dissimilar steps regarding tree individual node.
        a. Choose randomly r from R
        b. Select optimal between f.
        c. Split node.
2. When the creation of T trees is carried out, examples of test data will be sent to individual tree & allocation of class tag on basis of majority votes will be carried out
End
```

During the generation of neighbor solutions, dissimilarity among current& fresh neighbor-solution managed by means of a coefficient, α. The change from current point is multiplied by α during building new neighbors. Coefficient α is in the form of a sine function

α = 121 + siniθπNneigh Eq 1

Here i am the index of neighbor, Nneigh is entire number of neighbor solutions which is being produced at each iteration, & θ is a parameter that controls the oscillation period of α.
An objective principle that intended based on sigmoid function given by

Sk=11+e−σk–kcenter×M Eq 2

fkx−fk−Γxfk−Γx<δ Eq 3

Where δ is the ratio of alteration in objective-function value, Γ = ηM, and η is the fraction of the maximum iterations (M) by which the modify in the objective-function is evaluated. As per this stopping measure, if the enhancement over Γ generations is no longer than a threshold

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

(δ), continuation of further iterations can be in-effective& search should be terminated.

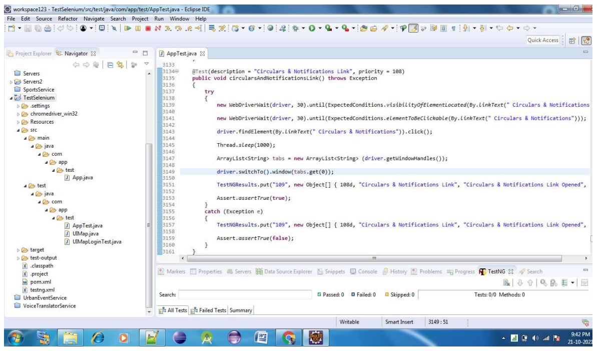
6. EXPERIMENTAL RESULTS

As previously stated, our ambition is to use machine-learning algorithm &TestNG framework to automatically test a Web-based application. We tested several powerful machine-learning models, including Support-Vector-Machine (SVM), Random-Forest &Random Integration, to find the best Machine-Learning algorithm capable of testing more precisely. To aid in successful evaluation of these replica's, we utilized machine-learning libraries.

TEST CASES

Positive Results

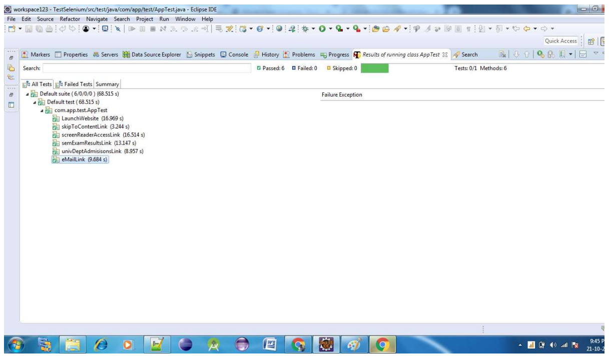
Load the website to be tested into the selenium web driver using the base URL, then write selenium commands to access specific features such as buttons, hyperlinks, test boxes, and so on to execute test cases. Positive outcomes will appear as binaries "1," as shown in the screenshotsbelow.



```
@Test(description = "Circulars & Notifications Link", priority = 108)
public void circularsAndNotificationsLink() throws Exception
{
    try
    {
        new WebDriverWait(driver, 30).until(ExpectedConditions.visibilityOfElementLocated(By.linkText(" Circulars & Notifications

        new WebDriverWait(driver, 30).until(ExpectedConditions.elementToBeClickable(By.linkText(" Circulars & Notifications")));

        driver.findElement(By.linkText(" Circulars & Notifications")).click();

        Thread.sleep(1000);

        ArrayList<String> tabs = new ArrayList<String> (driver.getWindowHandles());

        driver.switchTo().window(tabs.get(0));

        TestNGResults.put("109", new Object[] { 108d, "Circulars & Notifications Link", "Circulars & Notifications Link Opened",

        Assert.assertTrue(true);
    }
    catch (Exception e)
    {
        TestNGResults.put("109", new Object[] { 108d, "Circulars & Notifications Link", "Circulars & Notifications Link Opened",

        Assert.assertTrue(false);
    }
}
```

```
Passed: 0   Failed: 0   Skipped: 0   Tests: 0/0  Methods: 0
```

Fig-3 Test-Cases For Positive Results

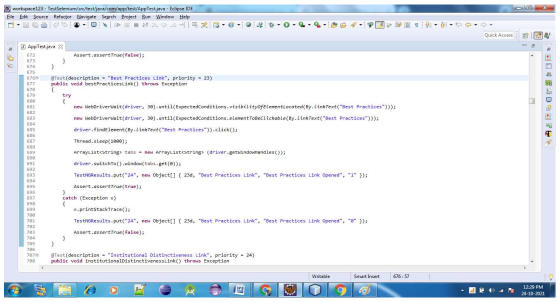


```
Passed: 6   Failed: 0   Skipped: 0   Tests: 0/1  Methods: 6

Default suite ( 6/0/0/0 ) (68.515 s)
  Default test ( 68.515 s)
    com.app.test.AppTest
      LaunchWebsite  (16.969 s)
      skipToContentLink  (3.244 s)
      screenReaderAccessLink  (16.514 s)
      semExamResultsLink  (13.147 s)
      univDeptAdmisisonsLink  (8.957 s)
      eMailLink  (9.684 s)
```

Fig-4 Test-Cases For Selenium

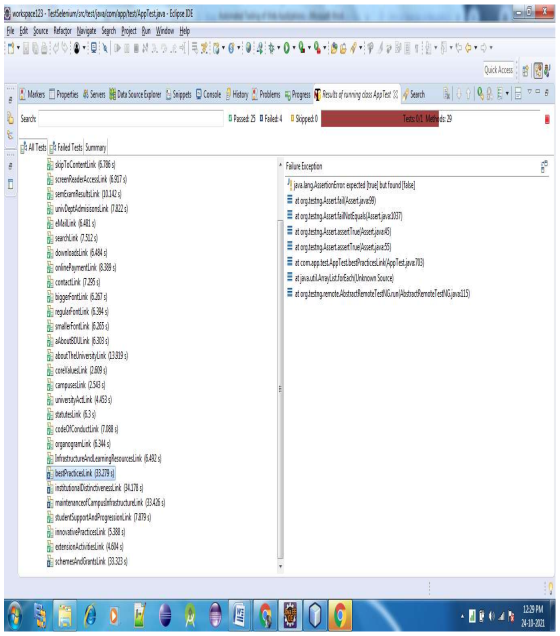
Negative Results

For negative cases the results will be in the form of binaries "0", kindly look at the below furnished screenshots for more details.



```
            Assert.assertTrue(false);
        }
    }

    @Test(description = "Best Practices Link", priority = 23)
    public void bestPracticesLink() throws Exception
    {
        try
        {
            new WebDriverWait(driver, 30).until(ExpectedConditions.visibilityOfElementLocated(By.linkText("Best Practices")));

            new WebDriverWait(driver, 30).until(ExpectedConditions.elementToBeClickable(By.linkText("Best Practices")));

            driver.findElement(By.linkText("Best Practices")).click();

            Thread.sleep(1000);

            ArrayList<String> tabs = new ArrayList<String> (driver.getWindowHandles());

            driver.switchTo().window(tabs.get(0));

            TestNGResults.put("24", new Object[] { 23d, "Best Practices Link", "Best Practices Link Opened", "1" });

            Assert.assertTrue(true);
        }
        catch (Exception e)
        {
            e.printStackTrace();

            TestNGResults.put("24", new Object[] { 23d, "Best Practices Link", "Best Practices Link Opened", "0" });

            Assert.assertTrue(false);
        }
    }

    @Test(description = "Institutional Distinctiveness Link", priority = 24)
    public void institutionalDistinctivenessLink() throws Exception
```

Fig-5 Test-Cases For Negative Results



```
Passed: 25   Failed: 4   Skipped: 0   Tests: 0/1  Methods: 29

skipToContentLink  (6.786 s)
screenReaderAccessLink  (6.917 s)
semExamResultsLink  (10.142 s)
univDeptAdmisisonsLink  (7.822 s)
eMailLink  (6.481 s)
searchLink  (7.512 s)
downloadsLink  (6.484 s)
onlinePaymentLink  (8.389 s)
contactLink  (7.295 s)
biggerFontLink  (6.267 s)
regularFontLink  (6.394 s)
smallerFontLink  (6.265 s)
aAboutBDULink  (6.303 s)
aboutTheUniversityLink  (13.919 s)
coreValuesLink  (2.609 s)
campusesLink  (2.543 s)
universityActLink  (4.453 s)
statutesLink  (6.3 s)
codeOfConductLink  (7.088 s)
organogramLink  (6.344 s)
InfrastructureAndLearningResourcesLink  (6.492 s)
bestPracticesLink  (33.279 s)
institutionalDistinctivenessLink  (34.178 s)
maintenanceofCampusInfrastructureLink  (33.426 s)
studentSupportAndProgressionLink  (7.879 s)
innovativePracticesLink  (5.388 s)
extensionActivitiesLink  (4.604 s)
schemesAndGrantsLink  (33.323 s)

Failure Exception
java.lang.AssertionError: expected [true] but found [false]
at org.testng.Assert.fail(Assert.java:99)
at org.testng.Assert.failNotEquals(Assert.java:1037)
at org.testng.Assert.assertTrue(Assert.java:45)
at org.testng.Assert.assertTrue(Assert.java:55)
at com.app.test.AppTest.bestPracticesLink(AppTest.java:703)
at java.util.ArrayList.forEach(Unknown Source)
at org.testng.remote.AbstractRemoteTestNG.run(AbstractRemoteTestNG.java:115)
```

Fig-6 Test-Cases For Selenium

Table 1 Accuracy Comparison Of SVM, Random Forest And Random Integration Algorithm.

ALGORITHM	50 Iteration's	100 Iteration's
RANDOM INTEGRATION	80	160
RANDOM FOREST	78	156
SVM	75	150

Table 2 Response Time Comparison Of SVM, Random Forest And Random Integration Algorithm.

ALGORITHM	50 Iteration's	100 Iteration's
SVM	7.2 milliseconds(ms)	17.8 milliseconds(ms)
RANDOM FOREST	7.0 milliseconds(ms)	15.2 milliseconds(ms)
RANDOM INTEGRATION	6.5 milliseconds(ms)	13.5 milliseconds(ms)

Table-2 displays the time spent testing an identical test case using the three testing strategies, while Table-1 displays the correctness of the three methods. Only the time spent developing testing scripts in preparation for the execution phase is included in these numbers. The execution time is indicated in milliseconds (ms). Table-2 shows that Random Integration Algorithm requires the least amount of time to run when compared to Support Vector Machine and Random Forest Algorithm. The Random Integration Algorithm takes 6.5 milliseconds in total to apply (ms). The Support-Vector-Machine (SVM) and Random Forest algorithms, respectively, take 7.2 milliseconds (ms) and 7.0 milliseconds (ms).

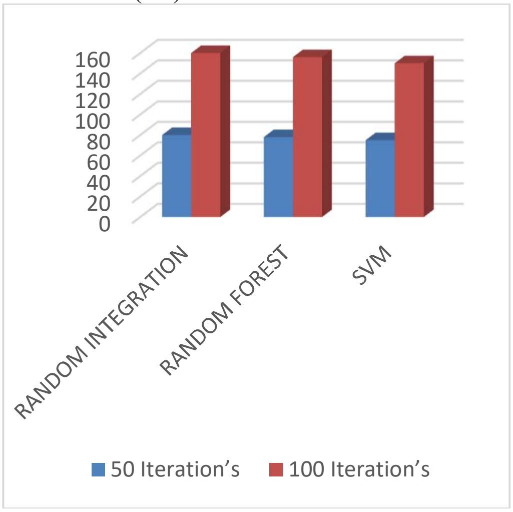


Fig 7 Accuracy Comparison

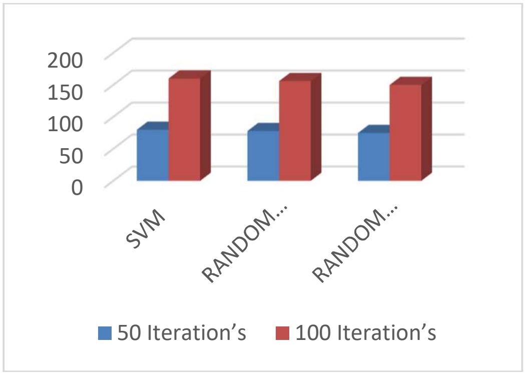


Fig 8 Response Time Comparison

On the other hand, the accuracy using Random Integrationalgorithm is 80 percentages at the 50th iterationwhereas the accuracy of Random-Forest &Support-Vector-Machine (SVM) algorithm is 78% & 75%. Thus, by using Random Integrationalgorithm can increase accuracy about approximately 5% more comparing with Random-Forest & Support-Vector-Machine (SVM) algorithm. Tabulated values accuracy and time-consumption is graphically represented in Fig 7 and Fig 8 which gives the clear understanding of all the three algorithms in performance wise.

7. DIFFERENCE FROM PRIOR WORK:

By Automating The Testing Framework Manual Testing Task Will Become Easy. IN Previous work there is no such arrangement.

8. CONCLUSION

The major goal of study is to develop a new software-based automation software-test framework to aid testers in automation software testing process. Projected framework can be utilized successfully to automate web application testing. When the Software-Under-Test (SUT) rapidly changing & regression testing is required to ensure that software version is stable, the TestNG framework is advantageous. Furthermore, software-testers or business end-users capable to do whatever they want while test scripts are running on SUT. Experiments undertaken to assess projected framework's efficiency by applying it to a case study. These tests have yielded promising results, implying that the full software-testing automation process could be automated to achieve 80% accuracy in less time. It can accomplish so much work at once that manual testing is almost impossible to complete. Furthermore, the suggested solution generates test-cases in a ordinary tabular format, which is both easier & tighter than writing them manually.

REFERENCES

[1] T. Kanstrén, "A Review of Domain-Specific Modelling, Software Testing," The Eighth International MultiConference on Computing in the Global Information Technology, 2013.

[2] A. Jain and S. Sharma, "An Efficient Keyword Driven Test Automation Framework for Web Applications," International Journal of Engineering Science & Advanced Technology, vol. 2, no. 3, pp. 600-604, 2012.

[3] V. Sangave and V. Nandedkar, "Generic Test Automation," International Journal of Science and Research (IJSR), vol. 4, no. 7, 2015.

[4] M. Sadiq and F. Firoze, "A Method for the Selection of Software Testing Automation Framework using Analytic Hierarchy Process," International Journal of Computer Applications (0975 – 8887), 2014.

[5] A. Cervantes, "Exploring the Use of a Test Automation Framework," IEEE Aerospace conference, 2009.

[6] Li, Yuan-Fang, Das, Paramjit K. and Dowe, David L.2014. Two decades of Web application testing—A survey of recent advances. Information Systems, Vol. 43, pp. 20–54.

[7] Z. Sun, Y. Zhang and Y. Yan, "A Web Testing Platform Based on Hybrid Automated Testing Framework," *2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, 2019, pp. 689-692, doi: 10.1109/IAEAC47372.2019.8997684.

[8] Y. Kravchenko, O. Makhovych, A. Yakymenko, S. Salkutsan, M. Tyshchenko and G. Smagulova, "Complex Dynamic Method of Web Applications Verification by the Criterion of Time Minimization," *2021 IEEE International Conference on Smart Information Systems and Technologies (SIST)*, 2021, pp. 1-5, doi: 10.1109/SIST50301.2021.9465904.

[9] M. Amouei, M. Rezvani and M. Fateh, "RAT: Reinforcement-Learning-Driven and Adaptive Testing for Vulnerability Discovery in Web Application Firewalls," in *IEEE Transactions on Dependable and Secure Computing*, doi: 10.1109/TDSC.2021.3095417.

[10] A. Sedaghatbaf, M. H. Moghadam and M. Saadatmand, "Automated Performance Testing Based on Active Deep Learning," *2021 IEEE/ACM International Conference on Automation of Software Test (AST)*, 2021, pp. 11-19, doi: 10.1109/AST52587.2021.00010.

[11] S. Karlsson, A. Čaušević and D. Sundmark, "Automatic Property-based Testing of GraphQL APIs," *2021 IEEE/ACM International Conference on Automation of Software Test (AST)*, 2021, pp. 1-10, doi: 10.1109/AST52587.2021.00009.

[12] S. Yu, C. Fang, Y. Yun and Y. Feng, "Layout and Image Recognition Driving Cross-Platform Automated Mobile Testing," *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, 2021, pp. 1561-1571, doi: 10.1109/ICSE43902.2021.00139.

[13] Y. Zheng *et al.*, "Automatic Web Testing Using Curiosity-Driven Reinforcement Learning," *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, 2021, pp. 423-435, doi: 10.1109/ICSE43902.2021.00048.

[14] A. Martin-Lopez, S. Segura, C. Muller and A. Ruiz-Cortes, "Specification and Automated Analysis of Inter-Parameter Dependencies in Web APIs," in *IEEE Transactions on Services Computing*, doi: 10.1109/TSC.2021.3050610.

[15] T. Rangnau, R. v. Buijtenen, F. Fransen and F. Turkmen, "Continuous Security Testing: A Case Study on Integrating Dynamic Security Testing Tools in CI/CD Pipelines," *2020 IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC)*, 2020, pp. 145-154, doi: 10.1109/EDOC49727.2020.00026.

[16] H. Zhu, "Software Testing as a Problem of Machine Learning: Towards a Foundation on Computational Learning Theory," *2018 IEEE/ACM 13th International Workshop on Automation of Software Test (AST)*, 2018, pp. 1-1.

[17] A. Arrieta, J. Ayerdi, M. Illarramendi, A. Agirre, G. Sagardui and M. Arratibel, "Using Machine Learning to Build Test Oracles: an Industrial Case Study on

Elevators Dispatching Algorithms," *2021 IEEE/ACM International Conference on Automation of Software Test (AST)*, 2021, pp. 30-39, doi: 10.1109/AST52587.2021.00012.

[18] B. Swathi and H. Tiwari, "Test Automation Framework using Soft Computing Techniques," *2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)*, 2021, pp. 1-4, doi: 10.1109/ICAECT49130.2021.9392602.

[19] J. Kahles, J. Törrönen, T. Huuhtanen and A. Jung, "Automating Root Cause Analysis via Machine Learning in Agile Software Testing Environments," *2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST)*, 2019, pp. 379-390, doi: 10.1109/ICST.2019.00047.

[20] Kumar, K. K., Srikanth, B., Kasiviswanadham, Y., Subbarao, C. D., Indira, D. N. V. S. L. S., & Pratap, N. L. (2021). The importance of light-weight encryption cipher in restricted IoT systems to make intelligent technology safer for devices. *Applied Nanoscience*, 1-11.

[21] Kumar, K. K., Chaduvula, K., & Markapudi, B. R. (2020). A Detailed Survey on feature extraction techniques in image processing for medical image analysis. *European Journal of Molecular & Clinical Medicine*, *7*(10), 2020.

[22] Indira, D. N. V. S. L. S., Babu, C. S., Kumar, K. K., & Rao, C. V. (2021). A comprehensive review on sentiment analysis techniques and machine learning libraries in image processing. *Annals of the Romanian Society for Cell Biology*, 4260-4267.

[23] Kumar, K. K. (2021). An Efficient image classification of malaria parasite using convolutional neural network and ADAM Optimizer. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, *12*(2), 3376-3384.

[24] K. K. kumar, E. Ramaraj and D. N. V. S. L. S. Indira, "Data Fusion Method and Internet of Things (IoT) for Smart City Application," *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, 2021, pp. 284-289, doi: 10.1109/ICICV50876.2021.9388532.

[25] Kumar, K. K., Ramaraj, E., & Geetha, P. (2021). Multi-sensor data fusion for an efficient object tracking in Internet of Things (IoT). *Applied Nanoscience*, 1-11.

[26]V. Chandra S.S, and A. Hareendran S, “Artificial Intelligence and machine learning,” PHI learning Private Limited, Delhi 110092, 2014