Open Access
Full Length Article

ORIGINAL ARTICLE

IMPROVING THE USABILITY TESTING: FUZZY BASED EXPERT SYSTEM FOR TEST CASE GENERATION ON WEB GRAPHICAL USER INTERFACE

Kinza Ijaz[1], Saima Munawar[2], Nasir Naveed[3]

[1,2,3] Department of Computer Science, Virtual University of Pakistan, Lahore

ABSTRACT

Usability testing (UT) technique is used to evaluate the user-friendliness of a website or of its interface without involving the actual users of the website. UT is performed either manually or by using an automated tool. The manual process of usability testing is time-consuming and costly. Manual work requires more resources (testers) and there is a considerable chance to get inconsistent results. The objective of this research is to improve the efficiency and reliability of test cases' (TC) generation whereas, the testing process is carried using automatic testing tools. Automated testing (AT) can be efficient and can provide accurate results. There are many automated tools available for software testing, with limited availability for TC automation. In this research, the systematic literature review (SLR) was conducted to find out the gap(s) in existing AT and challenges in TC generation. Secondly, the survey was conducted for identifying the main issues faced by different local testers during the process of generating TC manually. Fuzzy logic expert system was used to generate TC according to the selected suitable test cases. Fuzzy logic can emphasize non-probabilistic, uncertainty issues and multi-valued logic. The Data analysis was performed for the login page or registration page code and test cases were generated based on the GUI events through fuzzy logic. The system separated the keywords, attributes, and conditions from data analysis code and the output was displayed in the form of test cases. The comparative analysis was performed between manual TC generation processes with the fuzzy-based expert system for evaluation. The evaluation results obtained by statistical analysis showed that the proposed system is more efficient and reliable to generate test cases than the manual system.

Keywords: Automated testing, Artificial intelligence, Fuzzy system, Software Testing, Software Test cases, Usability Testing

Author`s Contribution
[1,2,3]Manuscript writing, Data analysis, interpretation, Conception, synthesis, planning of research, Interpretation and discussion, Data Collection

Address of Correspondence
Saima Munawar
saima.munawar@vu.edu.pk

Article info.
Received: Oct 19, 2018
Accepted: June 11, 2019
Published: June 30, 2019

Cite this article: *Ijaz K, Munawar S, Naveed N. Improving the Usability Testing: Fuzzy Based Expert System for Test Case Generation on Web Graphical User Interface. J. Inf. commun. technol. robot. appl.2019; 10(1):52-67*

Funding Source: *Nil*
Conflict of Interest: *Nil*

INTRODUCTION

Software testing is an important part of software development. The software can be tested either manually or by using an automated process. In manual testing, all tasks are performed by the tester, but in automatic testing, system is used to generate the report according to the required testing. Manual testing is a more hectic and expensive process as compared to automated testing. Manual testing requires a lot of effort from testers and there is considerable chance to get inaccurate results. In automated testing, different tools are used to automate the testing process that provides accurate output in less time. The AT preference is increasing because it is easy

and less expensive [1]. Various types of testing, such as correctness testing, performance testing, reliability testing, and security testing are tested through automated tools [2]. It is observed the 80% of errors in software can be removed by using testing tools such as CAPBAK, SMARTS, and EXDIFF, which are used for regression testing [3].

Many techniques of artificial intelligence (AI) are used for automated testing, it notes only reduces the cost but also improves the quality and reliability of [4]. Different tools and techniques are available for automating the test case generation process such as LEIRIOS Smart Testing technique, which was used for automation of test case generation [5]. A genetic algorithm with a mutation and a fitness function is also used to calculate the population of test cases [6]. Coded User Interface (UI) tool is used to automate the software testing, for example, Visual Studio Team Server (VSTS 2010). The random function such as Random r = new Random (); is used for test case generation [6]. The SLR was conducted to emphasize the existing automating techniques and current challenges that are faced during testing. There are many existing techniques for automating the test case generation process, but they do not provide accurate results as described in the methodology section ([7]; [8]; [9];[10]). Therefore, this research aims to improve the efficiency and reliability of test case generation by using fuzzy logic. The basic research question is "how can we improve the test case generation of usability testing on the web GUI interface scenario?". A survey was conducted on "Automation of software testing" from different local testers to support our proposed work. Data analysis was performed based on events of usability testing. The fuzzy-based expert system was designed in ASP.net language. Fuzzy logic is a technique of reasoning that resembles human reasoning. The fuzzy logic technique intimates the way of decision making. It contains all intermediate possibilities between digital values YES or NO. In this research, the fuzzy linguistics rules were generated for test cases. The inference model was used to examine the rule generation. A suitable selection of test cases has been preferred for performance evaluation. The manual process of test case generation is hectic and time-consuming. A tester has to write test cases multiple times for the same set of pages. In this research, test cases of common pages (such as registration or login page) code were generated through the expert fuzzy-based system. For example, login and registration are common pages on every website. Every time there is a change in the code, the tester has to write test cases for it, which is time-consuming. There is more chance of error in the manual process. Therefore the proposed expert system provides an efficient and flexible way for test case generation. MATLAB software was used to create membership function (MF). The system was developed using C# language to generate the test cases. The comparative analysis was performed between existing test case generation techniques with the proposed fuzzy-based test case generation process for evaluation.

The remainder of the paper is organized as follows. In Section 2, background and systematic literature review (SLR) are described. In section 3, the research methodology is presented. Section 4 presents a discussion of the results, while Section 5 concludes the paper and gives pointers to future work.

LITERATURE REVIEW

The testing procedure is used for the removal of software defects. Testing shows that the application properly implemented its planned work [11]. The quality of the software is measured and enhanced by software testing. Recognition, avoidance, demonstration, and increase in efficiency are the main objectives of software testing [12]. Software testing is performed in two ways, manually and by using automated testing tools.

In manual testing, test cases are written manually by the testers and implemented for error identification. To perform manual testing, a tester has to be trained, expert and open-minded. Manual testing of a large system is difficult to perform. The author claimed that the manual process of software testing is a time-consuming task and tedious process. The involvement of a large number of human resources is required in manual testing [13]. Testers use different testing techniques to achieve high coverage and efficiency of the software. The unit testing is the best technique for achieving high-quality software products and to get 100% coverage at this level. In embedded software, it is possible to use dynamic symbolic implementation methods and also achieve the

best quality result in coverage standards. It is also possible to design automated tools for cloud service [14]. AT increases coverage; decreases the effort of testing as compared to manual testing. The automated testing process is also used for the testing of business process management (BPM) applications and to reduce the test scripts' work creation and for automatic test case generation of BPM models. The functional test was the main focus that contains objectives such as from the flow analysis implementation path were recognized and the initial code was produced which works with Selenium-Cucumber Web application testing tools. As all the possible paths are analyzed the efficiency and coverage tests were enhanced [9]. The Activity flow graph model was used and this graph was derived from the event-flow graph. The input domain of GUI applications was described by an event flow graph. The Event-generation graph contains vertices, edges and initial vertices. The direction edges are also called follow-show, which shows the relationship between edge1 to edge 2. The experiment result shows that the test generation method is reliable [15]. As web applications become interesting, the complexity of web applications has also increased. The syntax model has developed for test cases. PSU-tep system was used in this research for the generation of test cases, but it is also mentioned in the further work to we need to develop more automatic tool [16].

In GUI testing, the performance of newly developed software is estimated. The automation of test case generation using automation tool is the most concentrated by researchers. A fuzzy model has been introduced to predict the usability of test cases for GUI. The usability testing has categorized as very low, low, intermediate, very high and high. The usability level of test cases and coverage standards are increased by using efficient computing methods. The output of the experiment was authenticated and acceptable [7].

Automation of software testing strictly depends on the system under test (SUT) and has limited functionality. Robotic Process Automation (RPA) was used, in which the desired result is achieved by emulating user actions within a graphical user interface. It provides a more reliable way for software test automation[17]

A newly improved method was introduced for GUI elements and components of already developed applications. The proposed method was fully automated and uses an artificial neural network to deal with duplicate test cases. A comparative analysis of different automated testing techniques was performed [18]. Various pillars of artificial intelligence (AI) have been discussed which are being used for software testing. It shows the future benefit of using AI and software testing. The result shows that the use of AI is useful in software testing and future AI-driven testing leads to a new era of Quality assurance (QA) [19].

There are many automation testing tools available for testing of the web applications. These automated testing tools save the testing time, cost and human effort. Comparative analysis of commercial and open-source web automation testing tools was performed [20]. The author described that unit testing is widely automated but system level is difficult to automate. If testing performs for Graphical user interface (GUI), then it's more difficult to automate system-level testing. This paper presents the agile software development environment, challenges in automating the system level software testing and some possible solution to these challenges [21]. Test automation technique has been presented by which test execution effort reduced from 2 days to 1 hour only. The regression testing performed once in a week but by using this automated technique it performs 4 times a week. The author claims that now they can transform the manual authors into a testing specialist [22].

The proposed methodology provides a static study and dynamic study of applications source code and executable files. This proposed method saves the human effort of about 61% as compared to manually creating test scripts [23]. All activities are discussed when someone performs software testing. Different automation testing problems are their solutions are also discussed in this research [24]. The black-box testing technique is used in which test cases are generated by using an Artificial intelligence (AI) planner. The test cases are generated from test objectives derived from UML (Unified Modeling Language) Class Diagram. The UML class diagram is used as a conceptual model for the tested system by which the test objective is obtained. The author suggested the use of the FUZed technique to generate Z specifications from UML models [25].

The new test scenario is generated by using specification and widgets extracted. In this method tester

only need to define the interaction of each widget which saves the testes effort of creating UI interaction scenarios. This method was implemented by using a GTA tool [26]. A fast and reliable technique has been presented for automating the test case generation process. In the proposed technique the manual input is not required for prioritization or the similarity tracking. The manual steps of selected test cases are used with already automated test cases and retrieved from test case repositories. Dynamic validation of the result presented after automating the manual test cases process. This result provides more tangible benefits in the future [27]. A clear review of android apps testing was presented. This paper presents the main trends, main methodologies, and challenges faced by android testing techniques [28]. Another author presented a method for user interface testing of a web-based application. By using specification and widgets extracted from the application new test scenario was generated. A GAT tool was used for Toshiba Software Development Vietnam (TSDV) and receives positive feedback[29].

There are many automation tools are available in the market such as JIRA, TFS, Test Director 8.0, Mantis Bug Tracker, Bugzilla, Selenium Web Driver, Badboy, Selenium IDE, IBM Rational Functional Tester, JUnit, NUnit, Cucumber. The study of requirements, planning of test cases, implementation of test, termination test and analysis of test are stages of software testing. They discussed levels of software testing such as unit testing, Incremental Integration Testing, Functional Testing, System Testing, acceptance testing, Beta Testing, End-to-end Testing, Regression Testing, Sanity Testing, Load Testing, Stress Testing, Performance Testing, Install/uninstall Testing, Compatibility Testing, Loop Testing and Recovery Test. The white box testing, black-box testing, and their types are also discussed [30]. A study described that white box testing is important for a tester to have complete knowledge about the source code. Risk analysis, development strategy, a proper test strategy was performed in white box testing techniques [31]. Manual testing and automated testing techniques are two testing techniques. Automated testing is further classified into four types such as correctness testing, performance testing, reliability testing, and security testing. Correctness testing has three forms such as white box testing, black-box testing, and grey-box testing [2]. Basic path flow was discussed in detail which contains a different graph flow, cyclomatic complexity, derived test cases and graph matrices [32].They described that black box testing is not just to correct the errors but expose more errors than white box testing [33]. Usability testing is a technique in which specific users can use a product to get a result efficiently in the identified context of usage. Effectiveness, efficiency, and satisfaction are attributes of usability testing [34].

- **Systematic Literature Review (SLR)**

The SLR includes the following steps to thoroughly investigate the "Automation of Software Testing" keyword.

Step 1: Search Terms and Results

In step 1, the following task has been performed.

"Automation of Software Testing" has been selected for the search term process based on background study. Springer, ACM and IEEE search engines have been chosen for SLR. There are 34 publications selected in IEEE, 131 selected publications in Springer and 206 selected publications in the ACM as shown in Figure 1. The selected searches found by using the "AND" operator. In Table 1, search terms and their results are shown.

Step 2: Search Process

The search process was selected based on an abstract, title, general study, and detailed study research. The searches are selected based on the inclusion and rejected based on exclusion criteria. The rules of inclusion and exclusion criteria are given below.

I. Only those articles are selected which are based on the "Automation AND software AND testing" keyword.
II. Only Springer, ACM and IEEE journals were selected for searching the keywords.
III. Those articles were not selected that were not discussed the "Automation of software testing from the perspective of test cases.

Table 1: Search Terms and Results

Sr. #	Search Term	Operator	IEEE	Springer	ACM
1	Automation of Software Testing	AND	34	131	206

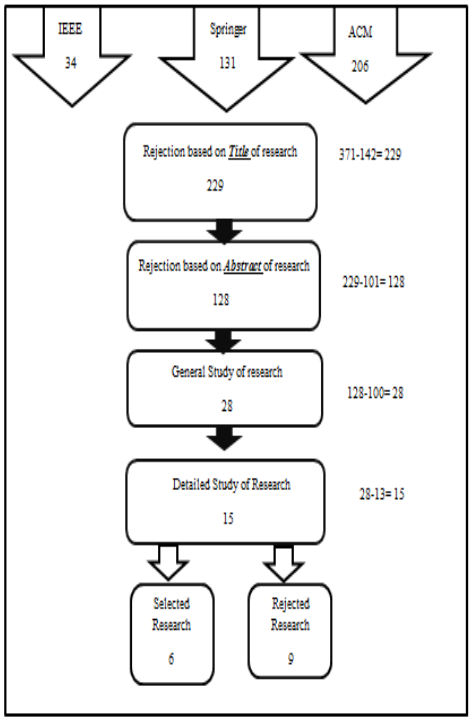

Figure 1. Systematic literature review of "Automation of Software testing" process

The inclusion and exclusion rules of the search process are as discussed below. There were a total of 371 search results from which 142 searches were rejected based on irrelevant abstract and 229 searches were accepted. A total of 142 search results was selected based on the abstract from which 101 results were rejected based on the title. The selected searches are based on the title was 142 of which 28 search results were selected based on the general study. The selected searches were based on the detailed study and 28 of which 15 searches were selected for detailed study. From these 9 searches were rejected and 6 searches were accepted.

- **Accepted primary studies**

The summary of the selected studies is given below.

This describes, how GUI and event-driven software testing took benefit from AI. Different AI techniques are used to automate the testing process, it not only reduced the cost but also the quality and reliability of testing were improved. The automated testing process has many advantages but has many problems as well that are still unsolved. The author claimed that manual testing can be replaced with an automated testing process by finding the error and increasing efficiency. Automated testing is restricted to software development and automation tools itself have no imaginations. Automated testing has been used for test generation of text cases, execution, control tests, to compare results with the standard output and report status development process. The author concluded that the proposed research presents the conditions in which AI is used for testing a GUI. It also described the advantages of using AI algorithms for software testing and GUI testing [4].

Software development testing plays an important role in finding bugs. The genetic algorithm was used in the testing process and also different testing techniques were discussed such as unit/Integration testing. QTP, load runners are the tools that were used for black-box testing, Evosuite, Jtest tools were used for white box testing. The author presented the genetic algorithm with a mutation and a fitness function to calculate the population of the test case. The formula a = b (c) +1 was used for mutation calculation and random function such as Random r = new Random (); was used for test case generation. In the first step, input was provided to system and test cases were generated by the proposed algorithm. The population was created for test cases and fitness functions were used to find the best test case. The mutation function mutes the value and provides the best output value as a result. The author used that random functions for single test case generations and the population was used for multiple test case generation. Genetic algorithms, fitness functions, and mutations were used for getting efficient output [6]. It described that as software complexities increase manual testing of software becomes more tedious and tough. During the testing of complex software, it analyzed automated testing that plays an important role. It can save testing time as well as provide better utilization of resources. A technique was introduced for automating the software testing process using Coded User Interface (UI) tool. The author represented a methodology of automation testing in which he discussed the general

format of a testing framework, which was cleared and handled under test software applications. Then the test was executed and the result was reported by using linear and structured scripted techniques. The coded UI automation tool in Visual Studio Team Server (VSTS) 2010 was also introduced. A generic framework presented in the Code UI tool and by using automation metrics result was observed. The author concluded that the introduced tool was beneficial for automating the testing process with wide test reporting, it can save time and resources as well [8].The development of software and its requirements are changed numerous times and because of these updating requirements are created errors in software. So they did perform manual and automation testing for removing these errors. As the use of software increase, they preferred automation testing because it was easy and less expensive. The author tackled the challenges of automation of the software testing process. The author discussed five units in which explained that how critical requirements are related to business plotted into formal specification language, by using formal requirements. The fitness function dependency graph (FFDG) and fitness function module were produced. A test task was established to design a test pattern and a template was generated dynamically. In that module, the intermediate test report was produced or the complete deliverable result took test reports. The author concluded that there was not a complete automation framework available in the literature, they are unable to compare our techniques with others. The complete experimental setup for evaluation of the proposed framework to show its effectiveness was required finishing framework and carrying out a full-scale empirical study will be covered in the future [1]. The LEIRIOS Smart Testing technique was used for the study of functional validation of the StarUML case subpart. The automation of testing of UML/MDA platform StarUML and by a case study the solution for automating the software testing was discussed. LEIRIOS Smart Testing was a behavioral UML model in which a diagram; object diagram and state machine was presented for designing such a test model. Test objective achieved from the UML behavioral test model and for automating the generation of test cases deductive engine was approved. This engine searches a path from the initial state to a test objective. Adapters and exporters were provided by the system for test cases generation. Different issues of the UML test model were discussed. The LEIRIOS Smart Testing solution was currently used in many applications [10].The reliability of software was improved by software testing. By automating the testing process, it becomes easy to test complex software efficiently. To design more capable software from the existing system assured the outcome some key features which were examined to relate testing to manufacture procedure. Different tools were discussed for addressing all these features. Test planning, coverage analysis, automatic regression, and system testing were the major parts that test by using Software Test Works (STW) tools. Nine tools were used which included in STW, CAPBAK SMARTS and EXDIFF and were mentioned regression test tools. SPECTEST, METATEST, and TDGEN were mentioned test planning tools [3]. The SLR emphasized that different testing techniques and tools were used to automate the testing process. The test case generation process can also be automated by using different AI techniques. The efficiency of software testing was increased by replacing automation testing with manual testing. Automation testing can save testing time as well as provide better use of resources.

RESEARCH METHODOLOGY

Automated testing is becoming an important part of software development as the use of software systems is increasing tremendously. Automated testing can be called time and cost-saving process. The reliability and efficiency of software can be improved by using automated testing techniques. In this section, five phases of the methodology have been presented. Initially, a survey has been conducted on the automation of software testing from different testers and identified problems of the existing system. In step 2, analysis of data gathered by conducting the survey has been performed. The analysis of GUI of registration or login page is also performed. In step 3, fuzzy model rules have been generated. In step 4, test cases have been generated. In step 5, test evaluation has been performed for checking performance and making a comparative analysis of manual or automated test case generation process. The methodology flow of research is as shown in Figure 2.

The detail of each phase is given below.

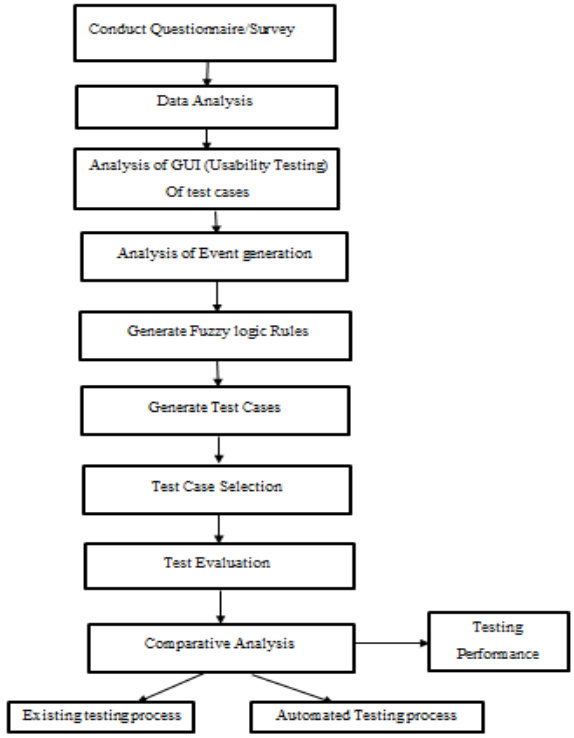

Figure. 2. Flow Diagram of Research Methodology

- **Questionnaire/Survey**

A survey has been conducted from different Software Quality Assurance (SQA) testers for getting information about the test case generation process. Different questions were asked about software testing from SQA testers, for example how do they perform testing? Which type of tools are used for testing? Which one is the easiest testing techniques? how did they ensure 100% testing and is it sufficient?. The questions are given in appendix A.

- **Analysis of Login interface (Usability Testing) of test cases**

The flow of data on the login interface has been shown by creating a data flow diagram. The code of login page has taken, keyword, attributes or conditions are separated from code for test case generation. The fuzzy rules, membership functions and test cases of code have been created. The detail is given below. Figure 3 shows how information flows on the login page. Firstly, get the reference to the model class in login.cs class. Then get user email and verify that the email field is not empty and format is correct. If the email is correct then getting the user password and check that password length is not exceeded to a given length, the password is in an encrypted format and the password field is not empty. The controller identifies the successful or unsuccessful login of users.

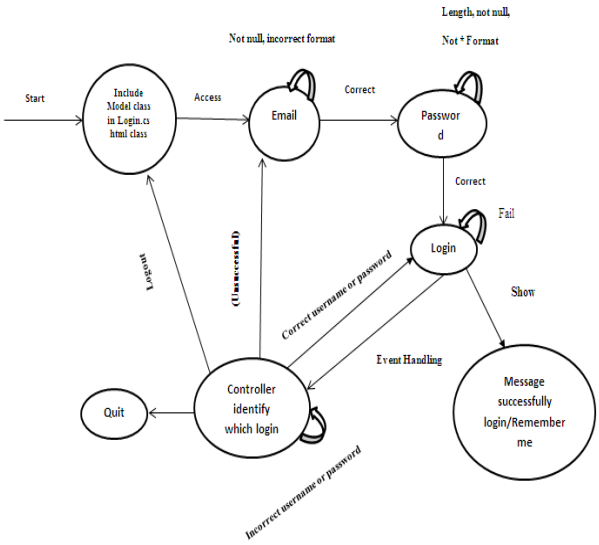

Figure 3. Data flow diagram of the login page

- **Login-view model page C# code**

The login page contains three fields i.e. user name, password and login button. The user enters the C# code to enter the code field. The keywords, attributes, and conditions have been separated from the code by system. The relevant keywords, attributes, and conditions are selected. Test cases are displayed to the user based on these keywords, attributes, and conditions.

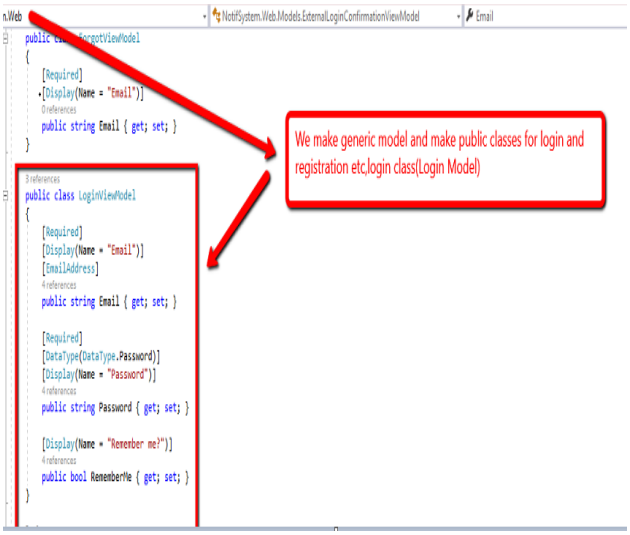

Figure 4. Login-view model page in C#

The output has been obtained according to generated fuzzy rules. The view model page code is shown in figure.4. The keywords, attributes, and conditions are separated from the code are given in Table 2. The dependencies between input and output variables of the login page are discussed in table 3.

Table 2: List of keywords, attribute, and conditions from login-view page code

Keywords	Attributes
Required	-
Display password (string)	-
@html.lLabel	Email/username
@html.Labelfor	password
@html.Checkboxfor(m->m.rememberme)	-
Display Email address/user name (string)	
Display remember me (bool)	-
ValidationMessage(m->m.Email/username)	-
ValidationMessage(m->password)	-
Labelfor(m->m.rememberfor)	-

Table 3: Dependencies between input and output variables

Label1+label2= submit
Email address+ password= submit
Label 1+label 2= label3
Email address+ password= checkbox remember me
Label 3=checkbox
Remember me= select/not select
Submit= Action result
Switch = submit
Submit also depend on switch cases.
Label 1+label 2= switch
Email+password=pass/fail/lockout/validation required
Label 1+label 2= Action Result
Email address+ password= Action Result

Steps for login

The details of the steps which are followed for login are discussed below. The working of each label and text box is discussed.

If label1 was an Email address.
If textbox1 was an Email address.
OR
If Email format is correct.
If Email format is incorrect.
If anyone option is selected.
Then action in step 1 was completed.
If step 1 was complete.
If label2 was password.
If textbox2 was password.
If textbox2 input length is Poor = 0
If textbox input length is medium=0-6
If textbox2 length is strong=8-15
If textbox2 length is very strong=20-25
If anyone option is selected.
OR
If password format= "*"
If password format is not "*"
If anyone option is selected.
Then action in step 2 was completed.
If step 2 was complete.
If the input type is submitted.
If submit attempt =0-5 then lockout
If Email and password is incorrect then require Validation
If Email correct and password is incorrect then require validation
If Email incorrect and password is correct then require validation
If Email and password are correct then login.
If anyone option is selected.
Then action in step 3 was completed.
If step3 was complete.
If checkbox1 for remember me.
If label3 for remember me
If Checkbox is not selected= 0
If checkbox is selected= 0-10
If anyone option is selected.
Then action in step 4 was completed.

- **Generate Fuzzy logic Rules for Login Page**

The fuzzy login rules have been generated for each test case of a login page. The "AND" operator is used for rule generation purposes. According to each fuzzy rule, triangle membership function is also created using MATLAB. In AND operator, both inputs are true then the result is true. If the input is wrong then output is false. For example, if the Email address is correct "AND" password length is correct then get access to log in. Table 4 shows Input 1 is an email address and input 2 is a password. If both email and password are correct then the user granted access to log in. If an email address is incorrect and the password is incorrect then it will not log in successfully.

Table 4: AND operator Example 1

Input 1	Operator	Input 2	Output
Email address= correct	AND	Password = correct	Login
Email address= incorrect	AND	Password = incorrect	Not login

The Formula used for count total number of fuzzy rules is Total input (n) mf ……eq.1

The detail of password length rules and MF are as discussed below. Password length is taken as input. The length of the password is categorized in 4 MF such as poor, medium, Strong and very strong. The value to each MF is assigned from 0 to 25. The pseudo-code of the password length is given below.

If password length is > "-1" AND <= "1"
then show message length = "poor".
Else If password length is >"1" AND <= "6"
then show message length = "medium".
Else If AND password length is > "6" AND <= "15"
then length = "strong";
Else If password length is >"15"and <= "25"
then length = "very strong".

Table 5 contains If Password length is equal to password maximum length then shows a message, strong or very strong password length or if the length is minimum then show a message, the length is poor or medium. The MF of password length is shown in Figure 5 on x-axis input variables. MF has been drawn according to the input variables as following.

Poor = (-1) -1
Medium = 1-6
Strong = 6-15
Very Strong =15-25

Table 5: Password Length Rule

Input 1	Operator	Input 2	Output
Password length maximum	AND	Password length minimum	Show message (length is poor, medium, strong, very strong)

Email addresses and passwords are taken as inputs. If both email addresses and passwords are correct then get access to log in. If anyone of these is incorrect then login is unsuccessful. Table 6 shows AND operator implemented on both inputs. If the Email address format is correct and the password is correct then login successfully. If anyone of these is incorrect then login is unsuccessful. The MF of Login successful/unsuccessful membership functions is shown in Figure 6.

If Email address ="correct" AND password = "correct" then login = "successful".
Else If Email address ="incorrect" and password = "incorrect" then login ="unsuccessful".
Else If Email address ="correct" AND password = "incorrect" then login = "unsuccessful".
Else If Email address ="incorrect" AND password = "correct" then login = "unsuccessful".

Table 6: Successful/unsuccessful login rule

Input 1	Operator	Input 2	Output
Email address= correct	AND	Password = correct	Login successful/unsuccessful

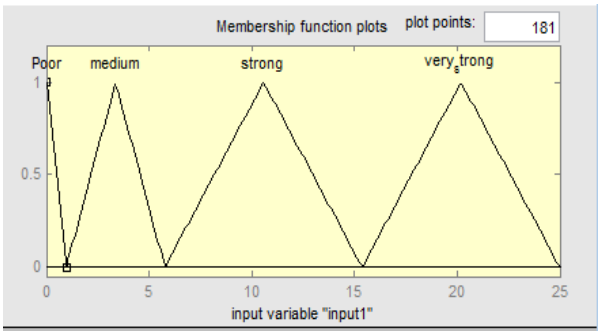

Figure 5. Password length Membership function

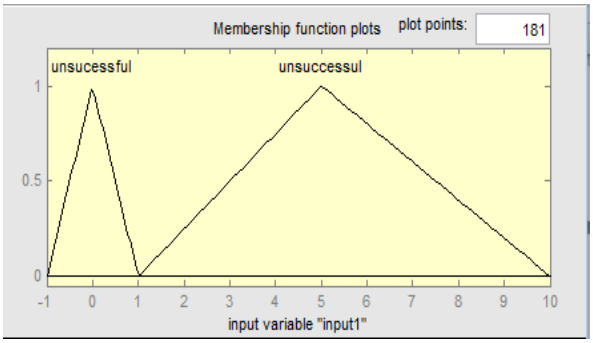

Figure 6. Login successful/unsuccessful membership function

- **Generate Test Cases of the Login page**

Test cases contain test scenario id, test scenario description, test case id, test case description, test steps, pre-conditions, test data, postconditions, expected results, and status. There are total 10 test cases of login page.

RESULTS AND DISCUSSION

A graphical user interface was built for a fuzzy-based test case generation. The system has been designed in visual studio software in C# language. The proposed system provides a facility to the users/tester to get the test cases of common pages such as the registration page or login page by creating accounts on the proposed system. The fuzzy MF also generated for each test case. Admin manages all users and all the working systems. The detail of the system is given below.

- **Admin interface**

Admin can directly login to the fuzzy-based expert system for test case generation by entering a valid email id, password and choose admin as the user type. After login, the admin home page can be accessed by admin as shown in Figure 7. On the admin home page, there is a record of all users who are registered in the system. The admin has the right to edit and update the record of each user. There is a list box under a user record from which the user can select a page whose test cases he/she can check. There is also a button for adding a new test case by click on the button, a user can get a page by which he/she can add new test cases. Figure 8 and Figure 9 show all test cases of the login page and registration page as created by admin. The admin has the right to edit or delete any test cases. There is a unique id assigned to each test case. The input, output, and status of test cases are also shown in the test case table. Admin views the MF of the password length and login status. In Figure 10 add new test cases page are shown. Admin enters the detail of new test cases by using this interface. There is a list box from which admin uses one page in which he/she can enter the new test case. SQL Server software is used for the database. All the records of registered users were stored in the database. The test case table was also stored in the database as shown in Figure 11. The test cases are fetched from the database when the admin views the test case.

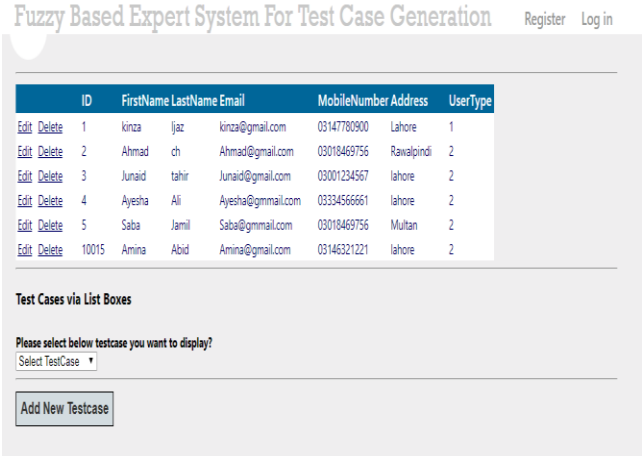

Figure 7. Admin home page

Figure 8. Registration text cases for admin

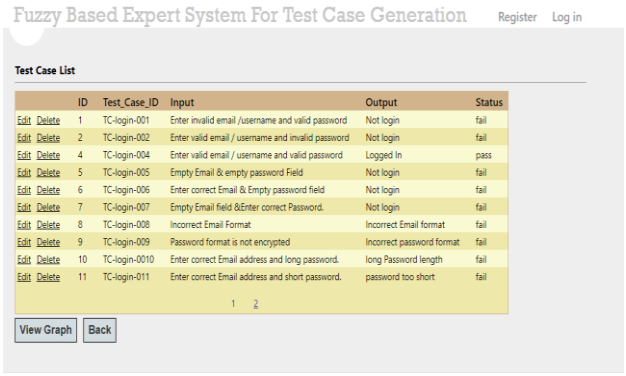

Figure 9. Login text cases for admin

DESKTOP-419JCH4\S....Testcase_Register

ID	Testcase_Regis...	Input	Output	Status
1	TC-reg-001	Left empty first ...	Not registered	Fail
2	TC-reg-002	Left empty last ...	Not registered	Fail
3	TC-reg-003	Left empty add...	Not registered	Fail
4	TC-reg-004	Left empty pho...	Not registered	Fail
5	TC-reg-005	Left empty ema...	Not registered	Fail
6	TC-reg-006	Left empty pass...	Not regstered	Fail
7	TC-reg-007	Enter empty co...	Not regstered	Fail
8	TC-reg-008	Enter encrypted...	Correct format	Pass
9	TC-reg-009	Enter password ...	Incorrect format	Fail
10	TC-reg-0010	Enter incorrect ...	Incorrect email ...	Fail
11	TC-reg-011	Enter long pass...	Password too l...	Fail
12	TC-reg-012	Enter short pas...	Password too s...	Fail
13	TC-reg-013	Enter different ...	Password and c...	Fail
14	TC-reg-014	Enter same pas...	Password and c...	Pass

Figure 10. Registration page test cases table in database

DESKTOP-419JCH4\S...SE - dbo.Testcase

ID	Test_Case_ID	Input	Output	Status
1	TC-login-001	Enter invalid e...	Not login	fail
2	TC-login-002	Enter valid ema...	Not login	fail
4	TC-login-004	Enter valid ema...	Logged In	pass
5	TC-login-005	Empty Email & ...	Not login	fail
6	TC-login-006	Enter correct E...	Not login	fail
7	TC-login-007	Empty Email fie...	Not login	fail
8	TC-login-008	Incorrect Email ...	Incorrect Email ...	fail
9	TC-login-009	Password form...	Incorrect passw...	fail
10	TC-login-0010	Enter correct E...	long Password l...	fail
11	TC-login-011	Enter correct E...	password too s...	fail
18	TC-login-019	Enter invalid E...	Not login	fail

Figure 11. Login page test cases table in database

- **User Main Interface**

The admin and tester can log in by entering a valid email address and password. The registration page contains the user's first name, last name, email, and password, confirm password, mobile number, and address field. The validation was applied to the email format. The format of the email must be correct otherwise user/tester cannot register. If the user/tester is already registered then he/she can directly login by entering registered an email address and password. There is an option of user type as admin and user/tester. The validation of the email field is shown in Figure 12. When the user enters the C# code of the login or registration page, then the condition, keyword, and attributes from the code are separated by system as shown in Figure 13. When the users click on the view test case then it is displayed to testers. In Figure 13, the login page code is entered by the user and Figure 14 shows the login test case table which is shown to the user. When the user enters registration code then the attributes, keywords, and conditions are separated and test cases as shown in Figure 15.

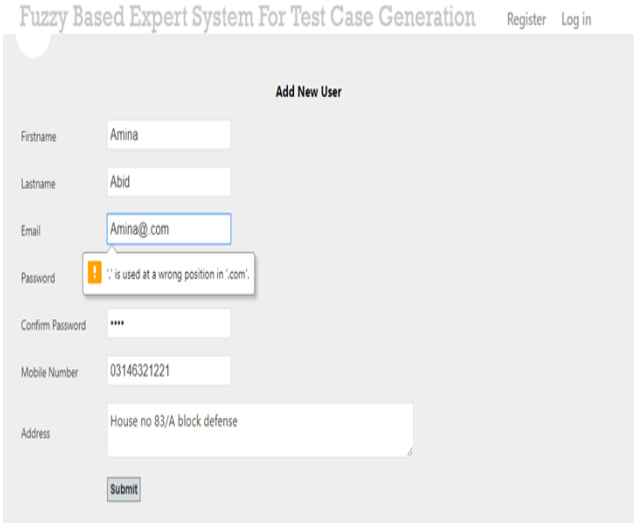

Figure 12. Validation on the Email address field

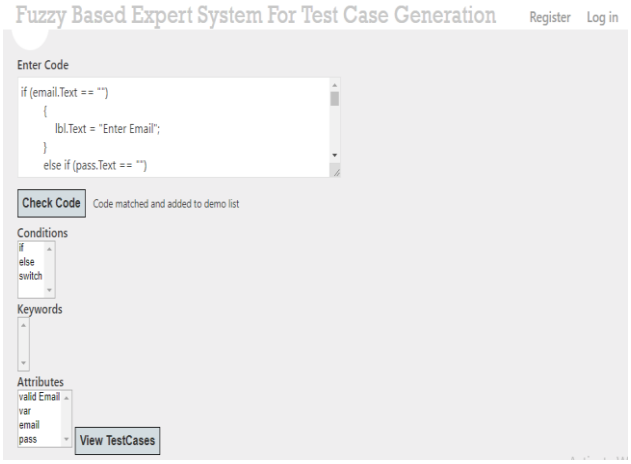

Figure 13. Login page code enter by user/tester

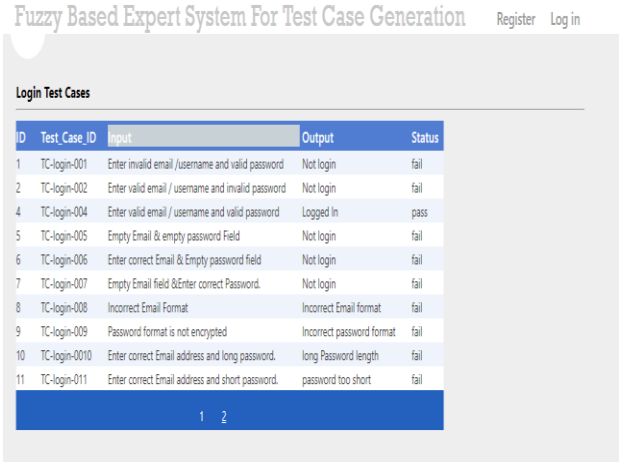

Figure 14. Login test cases table

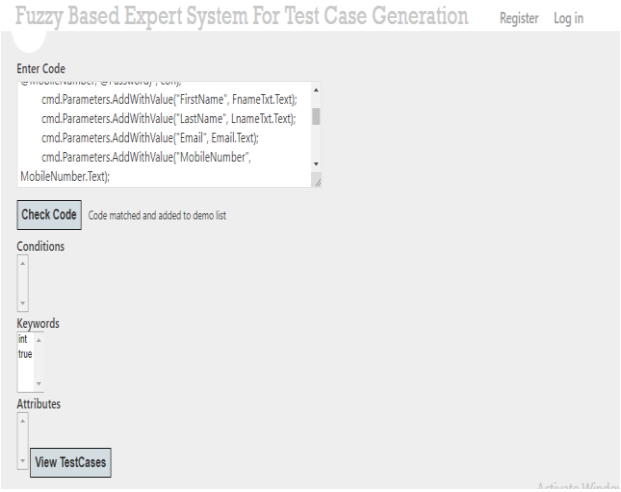

Figure 15. Registration page code enter by user

- **Comparison between Manual Testing System with Proposed System**

The system has scope to enter only C# code but it can be further enhanced and extended for other languages in the future. The C# code has been taken and it has done the task to generate test cases in both manually and automatically. The comparative study has been performed in both manual and automated test cases generation. The passed and failed test cases formulas are used to find the percentage of passed and failed test cases. After the manual testing, the same code has simulated by the proposed system. The system has separated keywords, attributes, and conditions by fuzzy rules. The test cases are generated according to code by the system. The interface is tested by using these test cases which is generated by the proposed system. Then compare the result of both processes by using the derivation formula [35] through the SPSS tool and find which one provides a more accurate result.

C# Code of login page taken as an example

The following C# code is used by both systems.

```
private void button1_Click (object sender, EventArgs e)
        if (txt_UserName.Text=="" ||
txt_Password.Text=="")
        {
           MessageBox.Show ("Please provide UserName
and Password");
           return;
        }
SqlCommand cmd = new SqlCommand("Select * from
tbl_Login where UserName=@username and
Password=@password", on);
           cmd.Parameters.AddWithValue ("@username",
txt_UserName.Text);
           cmd.Parameters.AddWithValue ("@password",
txt_Password.Text);
           con.Open();
           SqlDataAdapter adapt
= new SqlDataAdapter(cmd);
           DataSet ds = new DataSet();
           adapt.Fill(ds);
           con.Close();
           int count = ds.Tables[0].Rows.Count;
           //If count is equal to 1, than show frmMain form
           if (count == 1)
           {
              MessageBox.Show("Login Successful!");
              this.Hide();
              frmMain fm = new frmMain();
              fm.Show();
           }
           else
           {
              MessageBox.Show ("Login Failed!");
           }
```

Result of both systems

There is a total of 12 test cases implemented by both systems. From which 8 test cases passed and 4 failed. So a total of 67% of test cases passed and 33% test cases failed, the resulting analysis of manual testing is as shown in Figure 16. The percentage of passed and failed test cases are found by using the following formulas.

Test cases Passed % = **(**No. of Test cases Passed / Total no. of Test cases Executed) -- eq 2

Test cases Failed % = **(**No. of Test cases Failed / Total no. of Test cases Executed) --- eq3

There is a total of 12 test cases from which 10 test cases passed and 2 test cases failed, the resulting analysis of the proposed system is as shown in figure 17. A total of 83% of test cases passed and 17% test cases failed.

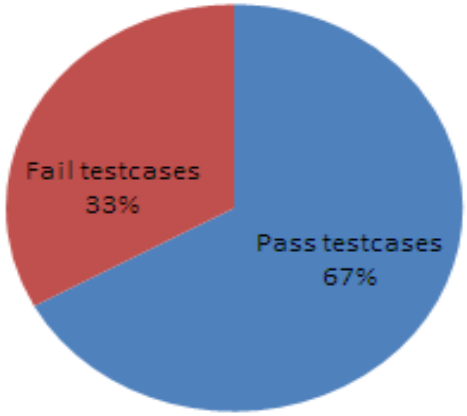

Figure 16. The resulting analysis of manual testing

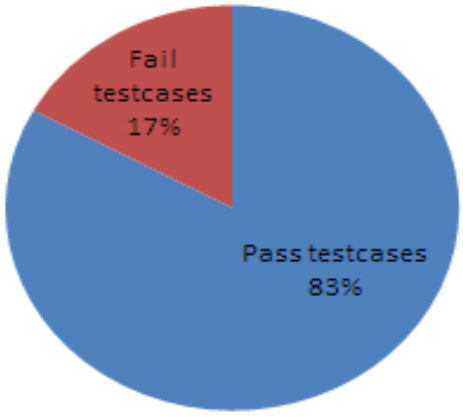

Figure 17. The resulting analysis of the proposed system

The standard deviation has been used for comparing the result of both systems. Test cases for two sample codes have been generated and tested by both systems. The result is shown in Table 7 and Table 8. Passed cases and failed cases of both systems have been taken and apply standard derivation by using the SPSS tool. In the given Table 7, Y1 represents percentage % of passing test cases of the proposed system and y2 represents the manual system passed test cases. In Table 8, Y1 represents failed test cases result of the proposed system and Y2 represents manual systems result. There are resultant data by passing test cases of the proposed and manual system known as a data point. There are two data points in sample code 1, 83 and 90 which got a total of 173. Then 173 are divided by the number of data points, in this case, result in a mean of 86.50. The result is determined by subtracting the value of the mean from each data point, resulting in -3.5 and 3.5. Then each of those values squared and has added together to get a result is 24.50. Then this result is divided by the value of N minus 1, which results is 24.50. The result of variance is calculated which get results in a standard deviation measure. The standard deviation of other values is measured by using the same procedure. The result after applying standard deviation on passing test cases result values are shown in Table 9 and failed test cases result in values as shown in Table 10. By comparative study, it concludes that the proposed fuzzy-based expert system provides more accurate results than a manual system. Manual testing is time taken process. A lot of testers/resource's time is wasted in writing test cases. By the proposed automated system, the same test case generation process is performed. It provides accurate results in less time. In the proposed system, users have an option to update, delete and add any test case if missing. The tester does not need to write test cases manually. He/she just enter the code and can get test cases according to code in a short time.

Table 7: Passed Test Cases of Manual System and Proposed System

Passed Case	Y1	Y2
Sample code 1	83	67
Sample code 2	90	58

Table 8: Fail Test Cases of Manual System and Proposed System

Filed Case	Y1	Y2
Sample code 1	33	17
Sample code 2	10	42

Table 9: Comparison between Passed Test cases result of both Systems by Deviation Descriptive Statistics

	N	Minimum	Maximum	Sum	Mean	Std. Deviation	Variance
Sample Code 1	2	83	90	173	86.50	4.950	24.500
Sample Code 2	2	58	67	125	62.50	6.364	40.500
Valid N (List wise)	2						

Table 10: Comparison between failed test cases result of both systems by deviation: Descriptive Statistics

	N	Minimum	Maximum	Sum	Mean	Std. Deviation	Variance
y1	2	10	17	27	13.50	4.950	24.500
y2	2	33	42	75	37.50	6.364	40.500
Valid N (List wise)	2						

CONCLUSION AND FUTURE RECOMMENDATIONS

Software testing is an important part of software development to produce high-quality software. There are different testing techniques used for software testing. The automation of software testing is the most efficient testing process as compared to the manual testing process. Automation of test case generation can reduce most of the tester's work. The SLR (Section 2) has been performed to emphasize the automation of software testing techniques. A survey has been conducted for analysis of the problem from different testers. In the proposed system Fuzzy rules are used for automating the test case generation process. Test cases are generated for common pages for example login and registration page. When these common pages are tested then tester writes test cases manually which require a lot of tester time and effort is wasted. The methodology of the fuzzy-based expert system for test case generation is very flexible. By using this expert system, the tester just enters the C# code of a page on the interface. Based on code keywords, attributes and conditions are separated and test cases are generated. Admin can create fuzzy rules for each test case by taking the code; Test cases are generated based on these rules. If there is any test case missing according to code then admin can add and manage it on their interface. The fuzzy triangle membership function is selected for representing each test case value graphically. The membership functions are created in MATLAB. Admin can also create a fuzzy triangle MF of each test case to check its performance on the interface. A fuzzy-based expert system is another step towards the automation of the test case generation process. The proposed system only generates test cases of login and registration page code written in C#. In the future, the expert system will generate test cases of all web pages of any language automatically by using a fuzzy-based system.

REFERENCES

1. M. M. Ali and T. K. Saha, "A proposed framework for full automation of software testing process," in 2012 International Conference on Informatics, Electronics & Vision (ICIEV), 2012, pp. 436-440.
2. A. A. Sawant, P. H. Bari, and P. Chawan, "Software testing techniques and strategies," International Journal of Engineering Research and Applications (IJERA), vol. 2, pp. 980-986, 2012.
3. E. Miller, "Advanced methods in automated software test," in Proceedings. Conference on Software Maintenance 1990, 1990, p. 111.
4. A. Rauf and M. N. Alanazi, "Using artificial intelligence to automatically test GUI," in 2014 9th International Conference on Computer Science & Education, 2014, pp. 3-5.
5. M. Alba, "A Test Generation Solution to Automate Software Testing," Journal Advances in Systems and Computer Science, ISSN, pp. 1657-7663, 2011.
6. G. M. Lodha and R. S. Gaikwad, "Search-based software testing with genetic using fitness function," in 2014 Innovative Applications of Computational Intelligence on Power, Energy, and Controls with their impact on Humanity (CIPECH), 2014, pp. 159-163.
7. T. Kushwaha and O. P. Sangwan, "Prediction of usability level of test cases for GUI based application using fuzzy logic," 2013.
8. P. Nagarani and R. VenkataRamanaChary, "A tool-based approach for automation of GUI applications," in 2012 Third International Conference on Computing, Communication and Networking Technologies (ICCCNT'12), 2012, pp. 1-6.
9. J. L. de Moura, A. S. Charão, J. C. D. Lima, and B. de Oliveira Stein, "Test case generation from BPMN models for automated testing of Web-based BPM applications," in 2017 17th International Conference on Computational Science and Its Applications (ICCSA), 2017, pp. 1-7.
10. F. Bouquet, C. Grandpierre, B. Legeard, and F. Peureux, "A test generation solution to automate software testing," in Proceedings of the 3rd international workshop on Automation of software test, 2008, pp. 45-48.
11. G. J. Myers, T. Badgett, T. M. Thomas, and C. Sandler, The art of software testing vol. 2: Wiley Online Library, 2004.

12. R. K. Chauhan and I. Singh, "Latest research and development on software testing techniques and tools," International Journal of Current Engineering and Technology, vol. 4, pp. 2347-5161, 2014.
13. R. Sharma, "Quantitative analysis of automation and manual testing," International journal of engineering and innovative technology, vol. 4, 2014.
14. C. Zhang, Y. Yan, H. Zhou, Y. Yao, K. Wu, T. Su, et al., "Smart unit: Empirical evaluations for automated unit testing of embedded software in industry," in Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice, 2018, pp. 296-305.
15. S. Mirshokraie, A. Mesbah, and K. Pattabiraman, "Atrina: Inferring unit oracles from GUI test cases," in 2016 IEEE International Conference on Software Testing, Verification and Validation (ICST), 2016, pp. 330-340.
16. J. Polpong and S. Kansomkeat, "Syntax-based test case generation for the web application," in 2015 International Conference on Computer, Communications, and Control Technology (I4CT), 2015, pp. 389-393.
17. S. Yatskiv, I. Voytyuk, N. Yatskiv, O. Kushnir, Y. Trufanova, and V. Panasyuk, "Improved Method of Software Automation Testing Based on the Robotic Process Automation Technology," in 2019 9th International Conference on Advanced Computer Information Technologies (ACIT), 2019, pp. 293-296.
18. Z. H. Cai, D. R. Li, M. Liu, Y. Shen, and K. Song, "Enhancing GUI automation testing using video," ed: Google Patents, 2018.
19. H. Hourani, A. Hammad, and M. Lafi, "The Impact of Artificial Intelligence on Software Testing," in 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT), 2019, pp. 565-570.
20. R. K. Lenka, U. Satapathy, and M. Dey, "Comparative Analysis on Automated Testing of Web-based Application," in 2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), 2018, pp. 408-413.
21. P. Aho and T. Vos, "Challenges in Automated Testing Through Graphical User Interface," in 2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 2018, pp. 118-121.
22. V. Garousi and E. Yildirim, "Introducing automated GUI testing and observing its benefits: an industrial case study in the context of law practice management software," in 2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 2018, pp. 138-145.
23. M. Iyama, H. Kirinuki, H. Tanno, and T. Kurabayashi, "Automatically Generating Test Scripts for GUI Testing," in 2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 2018, pp. 146-150.
24. Y. Labiche, "Test automation-Automation of what?," in 2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 2018, pp. 116-117.
25. A. Von Mayrhauser, R. France, M. Scheetz, and E. Dahlman, "Generating test-cases from an object-oriented model with an artificial intelligence planning system," IEEE Transactions on Reliability, vol. 49, pp. 26-36, 2000.
26. D. D. Tran, M. H. Nguyen, and N. H. Pham, "A method of Automated User Interface Testing for Windows-based Applications," VNU University of Engineering and Technology2018.
27. D. Flemström, P. Potena, D. Sundmark, W. Afzal, and M. Bohlin, "Similarity-based prioritization of test case automation," Software quality journal, vol. 26, pp. 1421-1449, 2018.
28. P. Kong, L. Li, J. Gao, K. Liu, T. F. Bissyandé, and J. Klein, "Automated testing of android apps: A systematic literature review," IEEE Transactions on Reliability, vol. 68, pp. 45-66, 2018.
29. S. Dhir and D. Kumar, "Automation Software Testing on Web-Based Application," in Software Engineering, ed: Springer, 2019, pp. 691-698.
30. S. H. Trivedi, "Software testing techniques," International Journal of Advanced Research in computer science and software engineering, vol. 2, pp. 433-438, 2012.
31. M. E. Khan and F. Khan, "A comparative study of white box, black box and grey box testing techniques," Int. J. Adv. Comput. Sci. Appl, vol. 3, 2012.
32. M. E. Khan, "Different approaches to white box testing technique for finding errors," International Journal of Software Engineering and Its Applications, vol. 5, pp. 1-14, 2011.
33. M. E. Khan, "Different approaches to the black-box testing technique for finding errors," International Journal of Software Engineering & Applications, vol. 2, p. 31, 2011.
34. T. Jokela, N. Iivari, J. Matero, and M. Karukka, "The standard of user-centered design and the standard definition of usability: analyzing ISO 13407 against ISO 9241-11," in Proceedings of the Latin American conference on Human-computer interaction, 2003, pp. 53-60.
35. H. Hassani, M. Ghodsi, and G. Howell, "A note on standard deviation and standard error," Teaching Mathematics and its Applications: An International Journal of the IMA, vol. 29, pp. 108-112, 2010.

REFERENCES

Survey Questions

Q.1. Which term you have mostly used for testing?

- Test Scenario
- Test cases

Q.2. Which type of testing is used for which type of application scenarios such as coding part, interface design, and database?

Q.3. How do you perform testing?

- Manually
- By using a tool

Q.4. which type of tool is used for software testing?

Q.5. which is the easiest testing technique?

Q.6. What is the most common bug in this easiest testing technique?

Q.7. Which technique is most commonly used for testing?

Q.8. How much time do you spend on each module of software testing?

Q.9. Is there any hole/problem which is not resolved by using this technique yet?

Q.10.Which testing technique is effected on time and cost of software project do you think?

Q.11. Which specification documents did you refer to write test cases?

Q.12. Did you have a situation where you did not have any documents (no requirement document, no use case and no design documents) and you were to write the test cases? How did you write the test cases?

Q.13. Do you really need to write SQL as a QA engineer?

Q.14. What is the most challenging situation you faced while applying to the test? How much testing is enough? How to declare the testing is enough/complete and on which stage?

Q.15. What are you going to do if there is no Functional Spec or any documents related to the system and developer who wrote the code does not work in the company anymore, but you have the system and need to test?

Q.16. How did you ensure that you cover 100% testing?

Q.17. How did you know, is it sufficient testing?

Q.18. Suggest me any technique for research.