

# ***SOFTWARE TESTING AND QUALITY ASSURANCE LAB***

## ***ETCS - 453***



Maharaja Agrasen Institute of Technology, PSP area,  
Sector - 22, Rohini, New Delhi - 110085

(Affiliated to Guru Gobind Singh Indraprastha  
University, New Delhi)

## **Introduction to Software Testing & Quality Assurance Lab**

### **1.1 Objective**

The objective of lab is to make the students aware about the existing methods of software testing and considering software quality. Course describes a wide range of techniques including Black Box testing, White box testing, mutation testing, slicing, test case coverage determination and Software quality factor, core components of quality, ISO standards and six sigma concepts etc. Many of the software testing techniques available are very expensive and time consuming. Therefore, the aim of the lab is to understand, which existing testing techniques are most effective for vulnerability detection, in order to provide software engineers guidelines for the selection of testing methods using software quality methods

The purpose of the laboratory is to evaluate and develop methods of testing software efficiently that aim on discovering security relevant software flaws along with considering core components of quality before the final product is deployed

## **1.2 Course outcomes**

**At the end of the course, a student will be able to:**

**C453.1:** Design and construct the adhoc test cases for different software module and manage a project using Quality standards from beginning to end.

**C453.2:** Generate test cases from software requirements using black box test processes for continuous quality improvement.

**C453.3:** Construct the test cases using white box software testing technique and ensure that the tests are be conducted according to the specification

**C453.4:** identify the needs of software test automation to solve specific problems alone or in team and define and develop a test tool to support test automation.

**C453.5:** Build an open source platform to develop a project/application for solving real world problem.

## Department of Computer Science and Engineering

### Rubrics for Lab Assessment

Rubrics		0	1	2	3
		Missing	Inadequate	Needs Improvement	Adequate
R1	Is able to identify the problem to be solved and define the objectives of the experiment.	No mention is made of the problem to be solved.	An attempt is made to identify the problem to be solved but it is described in a confusing manner, objectives are not relevant, objectives contain technical/ conceptual errors or objectives are not measurable.	The problem to be solved is described but there are minor omissions or vague details. Objectives are conceptually correct and measurable but may be incomplete in scope or have linguistic errors.	The problem to be solved is clearly stated. Objectives are complete, specific, concise, and measurable. They are written using correct technical terminology and are free from linguistic errors.
R2	Is able to design a reliable experiment that solves the problem.	The experiment does not solve the problem.	The experiment attempts to solve the problem but due to the nature of the design the data will not lead to a reliable solution.	The experiment attempts to solve the problem but due to the nature of the design there is a moderate chance the data will not lead to a reliable solution.	The experiment solves the problem and has a high likelihood of producing data that will lead to a reliable solution.
R3	Is able to communicate the details of an experimental procedure clearly and completely.	Diagrams are missing and/or experimental procedure is missing or extremely vague.	Diagrams are present but unclear and/or experimental procedure is present but important details are missing.	Diagrams and/or experimental procedure are present but with minor omissions or vague details.	Diagrams and/or experimental procedure are clear and complete.
R4	Is able to record and represent data in a meaningful way.	Data are either absent or incomprehensible.	Some important data are absent or incomprehensible.	All important data are present, but recorded in a way that requires some effort to comprehend.	All important data are present, organized and recorded clearly.
R5	Is able to make a judgment about the results of the experiment.	No discussion is presented about the results of the experiment.	A judgment is made about the results, but it is not reasonable or coherent.	An acceptable judgment is made about the result, but the reasoning is flawed or incomplete.	An acceptable judgment is made about the result, with clear reasoning. The effects of assumptions and experimental uncertainties are considered.

# SOFTWARE TESTING AND QUALITY ASSURANCE LAB PRACTICAL RECORD

PAPER CODE: ETCS – 453

S.No	EXPERIMENT NAME	DATE	MARKS					Total Marks	Signature
			R1	R2	R3	R4	R5		
1.	To determine the nature of roots of a quadratic equations, its input is triple of +ve integers (say x,y,z) and values may be from interval[1,100] the program output may have one of the following:- [Not a Quadratic equations, Real roots, Imaginary roots, Equal roots] Perform BVA and Robust Case Testing.								
2.	To determine the type of triangle. Its input is triple of +ve integers (say x,y,z) and the values may be from interval[1,100].The program output may be one of the following [Scalene, Isosceles, Equilateral, Not a Triangle].Perform BVA and Robust case Testing.								
3.	Consider the example of an app that classified Risk Exposure (RE) as High, Moderate or low on the basis of Risk Probability RP and Risk Impact (RI)...								
4.	Develop a complete limited entry decision table for the following decision situation: An airline offers flights only in Germany...								
5.	Create a test plan document for the application given in Experiment 4.								
6.	Study the Testing Tool: Win Runner								
7.	Study the Test Management Tool: QA Complete.								

8.	Automate the Test cases using Test Automation tool QA Complete.								
9.	Learn how to raise and report Bugs using Bug tracking tool (Bugzilla,Jira using QA Complete)								
10.	Study of any open source testing tool (Web Performance Analyzer/OSTA)								

# **EXPERIMENT – 1**

**AIM:** To determine the nature of roots of a quadratic equations, its input is triple of +ve integers (say x,y,z) and values may be from interval[1,100] the program output may have one of the following:- [Not a Quadratic equations, Real roots, Imaginary roots, Equal roots] Perform BVA and Robust Case Testing.

## **THEORY:**

### **Boundary Value Analysis:**

Boundary value analysis is one of the widely used case design techniques for black box testing. It is used to test boundary values because the input values near the boundary have higher chances of error.

Whenever we do the testing by boundary value analysis, the tester focuses on, while entering boundary value whether the software is producing correct output or not. Boundary values are those that contain the upper and lower limit of a variable. Assume that age is a variable of any function, and its minimum value is 18 and the maximum value is 30, both 18 and 30 will be considered as boundary values.

The basic assumption of boundary value analysis is, the test cases that are created using boundary values are most likely to cause an error.

There 18 and 30 are the boundary values that's why the tester pays more attention to these values, but this doesn't mean that the middle values like 19, 20, 21, 27, 29 are ignored. Test cases are developed for each and every value of the range.

There are a total of  $4n+1$  test cases in Boundary Value Analysis, where “n” is the number of input variables.

### **Input/Output Domain:**

Input : Three integers a, b, c.

Input Domain: Values from 1 to 100.

Output Type: One of the items printed from the output domain.

Output Domain:[Not a quadratic equation, Real roots, Imaginary roots, Equal roots].

### **Robustness Testing:**

Robustness is defined as the degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions. It is an Extension of boundary

value analysis. Any quality assurance methodology focused on testing the robustness of software. Used to describe the process of verifying the robustness (i.e. correctness) of test cases in a test process. When the extremes are exceeded with a value slightly greater than the maximum (max+) and a value slightly less than the minimum (min-). Robustness testing yields  $6n + 1$  test cases. Assume the min-, min, min+, nom, max-, max and max+ values, and repeat this for each variable.

### Input/Output Domain:

Input : Three integers a, b, c.

Input Domain: Values from 1 to 100.

Output Type: One of the items printed from the output domain.

Output Domain:[Not a quadratic equation, Real roots, Imaginary roots, Equal roots].

### BOUNDARY VALUE TEST CASES:

Test Case	Test Scenario	a	b	c	Expected Values	Actual Values	Pass/Fail
1	Quadratic equation	1	50	50	real roots	real roots	pass
2	Quadratic equation	2	50	50	real roots	real roots	pass
3	Quadratic equation	50	50	50	imaginary	imaginary	pass
4	Quadratic equation	99	50	50	imaginary	imaginary	pass
5	Quadratic equation	100	50	50	imaginary	imaginary	pass
6	Quadratic equation	50	50	1	real roots	real roots	pass
7	Quadratic equation	50	50	2	real roots	real roots	pass
8	Quadratic equation	50	50	99	imaginary	imaginary	pass
9	Quadratic equation	50	50	100	imaginary	imaginary	pass
10	Quadratic equation	50	1	50	imaginary	imaginary	pass
11	Quadratic equation	50	2	50	imaginary	imaginary	pass
12	Quadratic equation	50	99	50	imaginary	imaginary	pass
13	Quadratic equation	50	100	50	equal	equal	pass

**ROBUST CASE TEST CASES:**

Test Case	Test Scenario	a	b	c	Expected Values	Actual Values	Pass/Fail
1	Quadratic equation	0	50	50	Invalid Inputs	Invalid Inputs	pass
2	Quadratic equation	2	50	50	real roots	real roots	pass
3	Quadratic equation	50	50	50	imaginary	imaginary	pass
4	Quadratic equation	99	50	50	imaginary	imaginary	pass
5	Quadratic equation	100	50	50	imaginary	imaginary	pass
6	Quadratic equation	50	1	50	imaginary	imaginary	pass
7	Quadratic equation	50	2	50	imaginary	imaginary	pass
8	Quadratic equation	101	50	50	Invalid Inputs	Invalid Inputs	pass

**SOURCE CODE:**

```

#include<iostream>
using namespace std;

int main(){
    int a,b,c,d;
    cout<<"The quadratic equation is of the type  $a(x^2) + bx + c = 0$ "<<endl;
    cout<<"Enter a : ";
    cin>>a;
    cout<<"Enter b : ";
    cin>>b;
    cout<<"Enter c : ";
    cin>>c;

    d = (b*b)-(4*a*c);

    if((a<1)|| (b<1)|| (c<1)|| (a>100)|| (b>100)|| (c>100)){
        if(a==0)
            cout<<"Not a Quadratic Equation"<<endl;
    }
}

```



```

        else
            cout<<"Out of domain Input"<<endl;
    }
    else if(d==0)
        cout<<"Equal Roots"<<endl;

    else if(d>0)
        cout<<"Real Roots"<<endl;
    else //if d<0
        cout<<"Imaginary Roots"<<endl;

    return 0;
}

```

### **OUTPUT :**

```

The quadratic equation is of the type  $a(x^2) + bx + c = 0$ 
Enter a : 50
Enter b : 0
Enter c : 50
Out of domain Input

```

```

The quadratic equation is of the type  $a(x^2) + bx + c = 0$ 
Enter a : 50
Enter b : 1
Enter c : 50
Imaginary Roots

```

```

The quadratic equation is of the type  $a(x^2) + bx + c = 0$ 
Enter a : 50
Enter b : 100
Enter c : 50
Equal Roots

```

```
The quadratic equation is of the type  $a(x^2) + bx + c = 0$   
Enter a : 0  
Enter b : 50  
Enter c : 100  
Not a Quadratic Equation
```

```
The quadratic equation is of the type  $a(x^2) + bx + c = 0$   
Enter a : 1  
Enter b : 50  
Enter c : 99  
Real Roots
```

```
The quadratic equation is of the type  $a(x^2) + bx + c = 0$   
Enter a : 50  
Enter b : 101  
Enter c : 50  
Out of domain Input
```

### VIVA VOCE QUESTIONS

**Q1.What is the difference between Bug, Error and Defect?**

**Ans.** Bug: A bug is a mistake in the code that causes unexpected behavior in the program.

Error: An error is an incorrect result or unexpected behavior caused by a bug. Defect: A defect is a fault in the system caused by either a bug or an error. It is a deviation from the expected result.

**Q2. What is Boundary value analysis? Explain**

**Ans.** Boundary value analysis is a software testing technique used to identify defects in the software by testing values at the boundaries of the input domain. It involves testing the boundaries of the input data, such as minimum and maximum values, as well as values just inside and outside the boundaries. This method is used to find defects in the software, such as incorrect handling of input data, incorrect calculations, incorrect comparisons, etc. It is one of the most important techniques used in black box testing.

**Q3. What is Black Box testing technique?**

**Ans.** Black Box testing is a type of software testing technique that evaluates the functionality of a system or application without knowing its internal workings. This testing technique is based on the requirements and specifications of the system. Black box testing is used to validate the system's functionality and to ensure that the system meets user requirements. This type of testing is usually done by testers who are not familiar with the system's internal structure or code.

**Q4. How many numbers of test cases are there in boundary value analysis?**

**Ans.** The number of test cases in boundary value analysis depends on the size and complexity of the system being tested. Generally speaking, it is recommended to have at least three test cases for each boundary value.

**Q5. Explain the weakness of boundary value analysis.**

**Ans.** The main weakness of boundary value analysis is that it does not consider the behavior of the system for values between the boundaries. It only evaluates the system at the boundary values and does not consider any other input values. This means that errors or weaknesses in the system may not be detected if they occur for values between the boundary values. Additionally, boundary value analysis can be time-consuming and expensive, as it requires the execution of multiple test cases.

## **EXPERIMENT – 2**

**AIM:** To determine the type of triangle. Its input is triple of +ve integers (say x,y,z) and the values may be from interval[1,100].The program output may be one of the following [Scalene, Isosceles, Equilateral, Not a Triangle].Perform BVA and Robust case Testing.

**THEORY:**

### Boundary Value Test Cases:

Test Case	Test Scenario	a	b	c	Expected Output	Actual Output	pass/fail
1	Type of triangle	1	50	50	isosceles	isosceles	pass
2	Type of triangle	2	50	50	isosceles	isosceles	pass
3	Type of triangle	50	50	50	equilateral	equilateral	pass
4	Type of triangle	99	50	50	isosceles	isosceles	pass
5	Type of triangle	100	50	50	triangle is not valid	triangle is not valid	pass
6	Type of triangle	50	50	1	isosceles	isosceles	pass
7	Type of triangle	50	50	2	isosceles	isosceles	pass
8	Type of triangle	50	50	99	isosceles	isosceles	pass
9	Type of triangle	50	50	100	triangle is not valid	Triangle is not valid	pass
10	Type of triangle	50	1	50	isosceles	isosceles	pass
11	Type of triangle	50	2	50	isosceles	isosceles	pass
12	Type of triangle	50	99	50	isosceles	isosceles	pass
13	Type of triangle	50	100	50	Triangle Not valid	Triangle Not valid	pass

### Robust Case Test Cases:

Test Case	Test Scenario	a	b	c	Expected Output	Actual Output	pass/fail
1	type of triangle	0	50	50	Not a Triangle	Not a Triangle	pass

2	type of triangle	1	50	50	isosceles	isosceles	pass
3	type of triangle	2	50	50	isosceles	isosceles	pass
4	type of triangle	50	50	50	equilateral	equilateral	pass
5	type of triangle	99	50	50	isosceles	isosceles	pass
6	type of triangle	100	50	50	triangle not valid	triangle not valid	pass
7	type of triangle	101	50	50	Not a Triangle	Not a Triangle	pass
8	type of triangle	50	50	0	Not a Triangle	Not a Triangle	pass
9	type of triangle	50	50	1	isosceles	isosceles	pass
10	type of triangle	50	50	2	isosceles	isosceles	pass
11	type of triangle	50	50	99	isosceles	isosceles	pass
12	type of triangle	50	50	100	triangle not valid	triangle not valid	pass
13	type of triangle	50	50	101	Not a Triangle	Not a Triangle	pass

### **SOURCE CODE:**

```
#include<iostream>
using namespace std;

int main(){
    int side1, side2, side3;

    cout<<"Determining the type of Triangle"<<endl;
    cout<<"Enter side1 : ";
    cin>>side1;

    cout<<"Enter side2 : ";
    cin>>side2;

    cout<<"Enter side3 : ";
    cin>>side3;

    if((side1<1)||((side2<1)|| (side3<1))||(side1>100)||((side2>100)||((side3>100)))
        cout<<"Invalid Input"<<endl;
    else if ((side1 + side2 > side3 && side1 + side3 > side2 && side2 + side3 > side1) &&
        (side1 > 0 && side2 > 0 && side3 > 0)){
        if (side1 == side2 && side2 == side3)
```

```
        cout<<"Equilateral Triangle"<<endl;
    else if (side1 == side2 || side2 == side3 || side1 == side3)
        cout<<"Isosceles Triangle"<<endl;
    else
        cout<<"Scalene Triangle"<<endl;
    }
    else
        cout<<"Triangle not possible"<<endl;

    return 0;
}
```

### **OUTPUT:**

```
Determining the type of Triangle
Enter side1 : 50
Enter side2 : 50
Enter side3 : 50
Equilateral Triangle
```

```
Determining the type of Triangle
Enter side1 : 50
Enter side2 : 99
Enter side3 : 100
Scalene Triangle
```

### **VIVA VOCE QUESTIONS**

**Q1. What is equivalence partitioning explained with an example.**

**Ans.** Equivalence partitioning is a software testing technique used to reduce the number of test cases to a manageable number by dividing the input data of a software into classes of data from which test cases can be derived.

For example, if a software is designed to accept a user's age as input, equivalence partitioning would divide the input into two classes: valid ages (e.g. 18-100) and invalid ages (e.g. <18 and >100). The two classes can then be tested with a single test case each.

**Q2. What is Test bed and Test data ?**

**Ans.** Test bed is an environment specifically designed to allow testing of a system and its components. This environment usually consists of hardware, software, network components, databases and other necessary components.

Test data is the data set used for testing a system. It is the data used to check the accuracy, completeness, reliability and validity of a system. Test data is typically generated from actual data, but it can also be generated synthetically.

**Q3. Why does software have bugs?**

**Ans.** Software has bugs because of errors in the coding process. Software developers are human, and mistakes can and do happen. Bugs can also occur due to changes in the underlying hardware or software system, or due to changes in the programming language itself. Additionally, bugs can be introduced unintentionally through careless coding.

**Q4. How do you decide when you have 'tested enough'?**

**Ans.** When testing, it is important to balance the need to test thoroughly with the need to be efficient. This means deciding when you have tested enough depends on the goals of the project, the risks associated with the project, and the available time and resources. Generally, when the risk of not testing further is outweighed by the cost of additional testing, then it is time to stop testing.

**Q5. Describe the difference between validation and verification**

**Ans.** Validation is the process of making sure that a system meets the requirements and specifications of a customer or user. It is a process of evaluating a system to determine if it meets the pre-defined acceptance criteria.

Verification is the process of ensuring that a system is consistent and accurate, and that it works as expected. It is a process of checking the system to make sure it is working correctly and that it meets the desired requirements. Verification is a way of confirming the accuracy of the system.

**EXPERIMENT - 3**

**AIM:** Equivalence Class Partitioning In the lecture, we used the example of an app that classified Risk Exposure (RE) as High, Moderate, or Low on the basis of Risk Probability (RP) and Risk Impact (RI). Consider the following specification for such an app:

- 1) the app accepts two integers, RP and RI, as input,
  - 2) both RP and RI must be in the range [1, 5],
  - 3) if either input is not an integer, it is invalid and the app outputs “Invalid,
  - 4) if either input is an integer outside the range specified in (2), it is invalid and the app outputs “Out of Range”,
  - 5) given valid inputs, the app calculates RE as the product of RP and RI, and outputs: a. “High”, if RE is greater than 9b. “Low” if RE is less than or equal to 2c. “Moderate” if neither (a) nor (b) are satisfied
- i) Partition the domain of each parameter into equivalence classes, label the classes, and list them.
  - ii) Develop a set of test cases for the app to satisfy Each Choice Coverage of the equivalence classes. Indicate the equivalence classes covered by each test case and, as always, include the expected result. Notice the actual classification process is not adequately tested by your set of test cases.
  - iii) To better test the classification performed by the app, partition the output domain and develop additional test cases to cover any class not covered by your test cases in (ii).

## **Theory :**

Risk is the probability of a negative or undesirable outcome or event. Risk is any problem that might occur that would decrease customer, user, stakeholder perception of quality and/or project success.

### **Types of risks:**

There are 2 types of risks- Product Risk and Quality Risk.

#### **1. Product risk-**

When the Primary effect of a potential problem is on product quality, then the potential problem is called Product risk. It can also be called a quality risk. Example-A defect that can cause a system crash or monetary loss.

#### **2. Project Risk-**When the Primary effect of a potential problem is on the overall success of the project, those potential problems are called Project risk. They can also be called Planning risks. Example-Staffing issues that can delay project completion.



Not all risks are equally important. There is a number of ways we can classify the level of risk. The easiest way is to look at two factors-

1. Likelihood of occurrence of the problem. Likelihood arises from technical considerations.
2. Impact of the problem, if it occurs. Impact arises from business considerations.

### **What Is Risk-Based Testing?**

In risk-based testing, we use the risk items identified during risk analysis, along with the level of risk associated with each risk item to guide the testing. It is a type of software testing technique that is primarily based on the probability of the risk. Risk-based testing involves the following steps:

1. Assessing the risk based on software quality.
2. Frequency of use.
3. Criticality of Business.
4. Possible areas with defects, etc.

### **Characteristics Of Risk-Based Testing (RBT):**

Below are some characteristics of Risk-based testing (RBT)-

1. RBT strategy matches the level of test effort to the level of risk. The higher the risk, the more is the test effort. This applies to test execution as well as other test activities like test designing and implementation.
2. It matches the order of testing with the level of risk. Higher risk tests tend to find more defects or are related to more important areas in the application or maybe both. So higher the risk, we plan the tests earlier in the cycle-both in design and execution. This helps in building the confidence of business stakeholders as well.
3. By effort allocation and maintaining the order of testing, the quality risk gets systematically and predictably reduced. By maintaining a traceability matrix of tests vs risks and defects identified to risks, reporting the risk as residual risks make sense. This allows the concerned stakeholders to

decide when to stop testing i.e whenever the risk of continuing testing exceeds the risk of testing completion.

4.If schedule reduction requires scope reduction, it is easier to decide what needs to go out. It will always be acceptable and explainable to business stakeholders as risk levels are agreed upon.

5.To identify risks, we need to take inputs from various sources like –Requirements specifications, design specifications, existing application data, previous incident records, etc. However, if this information is not readily available, we can still use this approach by getting inputs from stakeholders for the risk identification and assessment process.

### **When To Implement Risk-Based Testing ?**

Risk-based testing approach is implemented in scenarios where-

- 1.There is time/resource or budget constraints. For example-A hotfix to be deployed in production.
- 2.When a proof of concept is being implemented.
- 3.When the team is new to the technology platform or to the application under test.
- 4.When testing in Incremental models or Iterative models.
- 5.Security testing is done in Cloud computing environments.

### **How Risk-Based Testing Is Implemented?**

Risk can guide testing in multiple ways but below are the major ones –

- 1.The effort is allocated by test managers proportionalto the risk associated with the test item.
- 2.Test techniques are selected in a way that matchesthe rigor and extensiveness required based on the level of risk of the test item.
- 3.Test activities should be carried out in reverseorder of risk i.e The Highest risk item should be tested first.
- 4.Prioritization and resolution of defects should be done as appropriate with the level of risk associated.
- 5.During test planning and control, test managers should carry outrisk controlfor all significant risk items. The higher the level of risk associated, the more thoroughly it should be controlled.

6. Reporting should be done in terms of residual risks. Example- Which tests have not run yet? Which defects have not fixed yet?

### Benefits of Risk-Based Testing (RBT):

By identifying and analyzing the risks related to the system, it is possible to make testing efficient and effective-

1. Efficient- RBT is efficient because you test the most critical areas of the system early in the cycle (the earlier the defect is detected the lower is the cost of solving those issues)
2. Effective- RBT is effective because your time is spent according to the risk rating and mitigation plans. You do not spend time on items and activities which might not be very important in the overall scheme of things.

### Solution

i)

RP	RI	Equivalence Class
RP<1	1<=RP<=5	Invalid
RP>5	1<=RP<=5	Invalid
1<=RP<=5	RP<1	Invalid
1<=RP<=5	RP>5	Invalid
Non Integer	1<=RP<=5	Invalid
Non Integer	RP<1	Invalid
Non Integer	RP>5	Invalid
RP<1	Non Integer	Invalid
RP>5	Non Integer	Invalid
1<=RP<=5	Non Integer	Invalid
1<=RP<=5	1<=RP<=5	Valid

II)

<b>RP</b>	<b>RI</b>	<b>Equivalence Class</b>	<b>Test Case RP</b>	<b>Test Case RI</b>	<b>Expected Output</b>
RP<1	1<=RP<=5	Invalid	0	3	Out of Range
RP>5	1<=RP<=5	Invalid	7	2	Out of Range
1<=RP<=5	RP<1	Invalid	3	-1	Out of Range
1<=RP<=5	RP>5	Invalid	4	23	Out of Range
Non Integer	1<=RP<=5	Invalid	23.4	3	Invalid
Non Integer	RP<1	Invalid	A	0	Invalid
Non Integer	RP>5	Invalid	Hello	17	Invalid
RP<1	Non Integer	Invalid	-45	3.2	Invalid
RP>5	Non Integer	Invalid	60	Xyz	Invalid
1<=RP<=5	Non Integer	Invalid	2	*	Invalid
1<=RP<=5	1<=RP<=5	Valid	2	3	Moderate

III)

<b>Test Case RP</b>	<b>Test Case RI</b>	<b>ExpectedOutput</b>
4	3	High
1	1	Low
3	2	Moderate

### **VIVA VOCE QUESTIONS**

**Q1. What is risk-based testing?.**

**Ans.** Risk-based Testing is the term used for an approach to creating a Test Strategy that is based on prioritizing tests by risk. The basis of the approach is a detailed risk analysis and prioritizing of risks by risk level. Tests to address each risk are then specified, starting with the highest risk first.

**Q2. How to perform risk based testing?**

**Ans.** 1.Make a prioritized list of risks.

2.Perform testing that explores each risk.

3.As risks evaporate and new ones emerge, adjust test effort to stay focused on the current crop.

**Q3. How to improve skills designing test cases and make sure high coverage rate?**

**Ans.** Test designing is successful when the requirements are analyzed and understood completely. To ensure 100% test coverage is achieved, you should not miss out on creating test cases for any requirements and from time to time we can check ourselves with the help of a traceability matrix.

**Q4. What's the difference between System testing and Acceptance testing?**

**Ans.** Acceptance testing checks the system against the "Requirements." It is similar to System testing in that the whole system is checked but the important difference is the change in focus:

System testing checks that the system that was specified has been delivered. Acceptance testing checks that the system will deliver what was requested. The customer should always do Acceptance testing and not the developer.

**Q5: How do you define a testing policy?**

The first step any organization needs to do is define one unique definition for testing within the organization so that everyone is of the same mindset.

**EXPERIMENT – 4**

**AIM:** Develop a complete limited entry decision table for the following decision situation:

An airline offers only flights in Germany and Europe. Under special conditions a discount is offered — a discount with respect to the normal airfare.

**Rules:**

- Passengers older than 18 with destinations in Germany are offered a discount at 20%, if the departure is not on a Monday or Friday- If the passengers stay at least 6 days at the destination, an additional discount of 10% is offered.
- For destinations outside Of Germany passengers are offered a discount Of 25%, if the departure is not on a Monday or Friday.
- Passengers older than 2 but younger than 18 years are offered a discount of 40% for all destinations.
- Children under 2 travel for free.

For each rule, design the test case

**Theory :**

A decision table is a brief visual representation for specifying which actions to perform depending on given conditions. The information represented in decision tables can also be represented as decision trees or in a programming language using if-then-else and switch-case statements.

A decision table is a good way to settle with different combination inputs with their corresponding outputs and is also called a cause-effect table. The reason to call cause-effect table is a related logical diagramming technique called cause-effect graphing that is basically used to obtain the decision table.

**Importance of Decision Table:**

- Decision tables are very much helpful in test design techniques.
- It helps testers to search the effects of combinations of different inputs and other software states that must correctly implement business rules.
- It provides a regular way of starting complex business rules, that is helpful for developers as well as for testers.
- It assists in the development process with the developer to do a better job. Testing with all combinations might be impractical.
- A decision table is basically an outstanding technique used in both testing and requirements management.
- It is a structured exercise to prepare requirements when dealing with complex business rules.
- It is also used in model complicated logic.

## **Advantages of Decision Table:**

- Any complex business flow can be easily converted into test scenarios & test cases using this technique.
- Decision tables work iteratively which means the table created at the first iteration is used as input tables for the next tables. The iteration is done only if the initial table is not satisfactory.
- Simple to understand and everyone can use this method to design the test scenarios & test cases.
- It provides complete coverage of test cases which helps to reduce the rework on writing test scenarios & test cases.
- These tables guarantee that we consider every possible combination of condition values. This is known as its completeness property.

## **Extracting Rules**

If all the conditions and actions from the scenario are extracted, you can summarize it as follows:

### **Conditions:**

- Destination (Germany, Europe)
- Passenger Age ( $\leq 2$ ,  $> 2 \ \&\& \ < 18$ ,  $> 18$ )
- Depart on Monday or Friday (Yes, No)
- Stay 6 days or more (Yes, No)

### **Actions:**

- Travel Free
- 0% discount
- 10% discount
- 20% discount
- 40% discount

Number of rules: 2 values \* 3 values \* 2 values \* 2 values = 24 rules

Now we dump the result in a simple table to visualize a combination of all these conditions.

## Scenarios

Airline Discount Policy	Rules																							
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Destination:	I	I	I	I	I	I	I	I	I	I	I	I	A	A	A	A	A	A	A	A	A	A	A	A
Passenger Age:	<=2	<=2	<=2	<=2	3-18	3-18	3-18	3-18	>18	>18	>18	>18	<=2	<=2	<=2	<=2	3-18	3-18	3-18	3-18	>18	>18	>18	>18
Depart Mon/Fri?	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N
Stay >= 6 Days?	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N
No Discount										X												X		
10% Discount	X		X		X		X		X		X		X		X		X		X		X		X	
20% Discount											X	X												
25% Discount															X	X			X	X			X	X
40% Discount					X	X	X	X									X	X	X	X				
Travel Free	X	X	X	X									X	X	X	X								

## Reduced Table

Airline Discount Policy	Rules												
	1	2	3	4	5	6	7	8	9	10	11	12	13
Destination:	--	I	I	--	--	I	I	A	A	A	A	A	A
Passenger Age:	<=2	3-18	3-18	>18	>18	>18	>18	3-18	3-18	3-18	3-18	>18	>18
Depart Mon/Fri?	--	--	--	Y	Y	N	N	Y	Y	N	N	N	N
Stay >= 6 Days?	--	Y	N	N	Y	Y	N	Y	N	Y	N	Y	N
No Discount				X									
10% Discount			X		X	X		X		X		X	
20% Discount						X	X						
25% Discount										X	X	X	X
40% Discount		X	X					X	X	X	X		
Travel Free	X												

## Preparing Decision Table



Now we can define the aging groups as shown in the following definition:

- Group1 :  $\leq 2$
- Group2 : 3-18
- Group3 :  $> 18$

It is simpler when we prepare a model to help make a decision. It would encapsulate all conditions:

- Destination: The destination of travel
- AgeGroup: The aging group as defined
- IsWeekDay: The condition for “Depart Mon/Fri?”
- StayMoreThan6: The condition for “Stay  $\geq 6$  Days?”

	Destination	Age	IsWeekDay	MoreThan6	Discount
Rule 1		Group1			100
Rule 2	Germany	Group2		Y	40
Rule 3	Germany	Group2		N	50
Rule 4		Group3	Y	N	0
Rule 5		Group3	Y	Y	10
Rule 6	Germany	Group3	N	Y	30
Rule 7	Germany	Group3	N	N	20
Rule 8	Europe	Group2	Y	Y	50
Rule 9	Europe	Group2	Y	N	40
Rule 10	Europe	Group2	N	Y	75
Rule 11	Europe	Group2	N	N	65
Rule 12	Europe	Group3	N	Y	35
Rule 13	Europe	Group3	N	N	25

**The final table looks like this:**

Destination	Age	IsWeekDay	MoreThan6	Discount
	Group1			100
Germany	Group2		Y	40
Germany	Group2		N	50
	Group3	Y	N	0
	Group3	Y	Y	10
Germany	Group3	N	Y	30
Germany	Group3	N	N	20
Europe	Group2	Y	Y	50
Europe	Group2	Y	N	40
Europe	Group2	N	Y	75
Europe	Group2	N	N	65
Europe	Group3	N	Y	35
Europe	Group3	N	N	25

**Result:**

For each rule test-case was designed and verified.

**VIVA VOCE QUESTIONS**

**Q1. Why we use decision tables?**

**Ans.** The techniques of equivalence partitioning and boundary value analysis are often applied to specific situations or inputs. However, if different combinations of inputs result in different actions being taken, this can be more difficult to show using equivalence partitioning and boundary value analysis, which tend to be more focused on the user interface. The other two specification-based techniques, decision tables and state transition testing are more focused on business logic or business rules. A decision table is a good way to deal with combinations of things (e.g. inputs). This technique is sometimes also referred to as a 'cause-effect' table. The reason for this is that there is an associated logic diagramming technique called 'cause-effect graphing' which was sometimes used to help derive the decision table.

**Q2 What are the disadvantages of Decision Table testing?**

**Ans.** The main disadvantage is that when the number of inputs increases the table will become more complex.

**Q3. Why Decision Table Testing is Important?**

**Ans.** Decision Table Testing is Important because it helps to test different combinations of conditions and provides better test coverage for complex business logic. When testing the behavior of a large set of inputs where system behavior differs with each set of inputs, decision table testing provides good coverage and the representation is simple so it is easy to interpret and use.

In Software Engineering, boundary value and equivalent partition are other similar techniques used to ensure better coverage. They are used if the system shows the same behavior for a large set of inputs. However, in a system where for each set of input values the system behavior is different, boundary value and equivalent partitioning technique are not effective in ensuring good test coverage. In this case, decision table testing is a good option.

This technique can make sure of good coverage, and the representation is simple so that it is easy to interpret and use.

This table can be used as the reference for the requirement and for functionality development since it is easy to understand and cover all the combinations.

**Q4. What are the advantages of Decision Table Testing?**

**Ans.** 1. When the system behavior is different for different inputs and not the same for a range of inputs, both equivalent partitioning, and boundary value analysis won't help, but a decision table can be used.

2. The representation is simple so that it can be easily interpreted and is used for development and business as well.

3. This table will help to make effective combinations and can ensure better coverage for testing

4. Any complex business conditions can be easily turned into decision tables

5. In a case we are going for 100% coverage typically when the input combinations are low, this technique can ensure the coverage.

**Q5: Why is Decision Table also called a Cause-Effect table?**

**Ans.** Decision table testing is a software testing technique used to test system behavior for different input combinations. This is a systematic approach where the different input combinations and their corresponding system behavior (Output) are captured in a tabular form. That is why it is also called as a Cause-Effect table where Cause and effects are captured for better test coverage.

**EXPERIMENT - 5**

**Aim** - Create a test plan document for Library Management System

## **THEORY: TEST PLAN DOCUMENT**

A Test Plan is a detailed document that describes the test strategy, objectives, schedule, estimation, deliverables, and resources required to perform testing for a software product. Test Plan helps us determine the effort needed to validate the quality of the application under test. The test plan serves as a blueprint to conduct software testing activities as a defined process, which is minutely monitored and controlled by the test manager.

### **Table of Contents :**

- 1) Introduction
- 2) Purpose
- 3) Scope
- 4) Testing Strategies
  - 4.1. Unit testing
  - 4.2 System & Integration testing
  - 4.3. Performance testing & Stress testing
  - 4.4. Acceptance Testing
- 5) features to be listed
- 6) Hardware requirements
- 7) Environment requirements
- 8) Test schedule
- 9) Risk & Mitigation
- 10) Tools

### **1. Introduction**

Every college/ school has their library both for teachers and students to use. The traditional system to manage them is either keeping track of them in a register or keeping track of a similar entry in computer . Its very time consuming . Online library management system helps in solving this issue .

### **2. Purpose**

The library management system is an online application for testing a librarian in managing book library in a university. This test plan document support the following objective :

- Identify existing project information of software that should be tested.
- List the recommended tools requirements.
- Recommend and describe the testing strategies to be employed.
- List the deliverable elements of the test activities.

### **3. Scope**

The system that is to developed provides the related information on students and system administration

- Creating a system administrator who will be the sole user managing the system on the backend.
- System admin can add/ delete/view/ edit the books
- Admin can add/delete/view/edit the books issued
- Admin can search for the books issued.

### **4. Testing Strategies**

The aim of the system testing process is to determine all the defects in the project.

#### **4.1 Unit Testing :**

In order to test a single module, we need to provide a complete environment and besides the module we could require

- The procedure belonging to other modules.
- Non local data structures that module accesses.
- A procedure to call the functions of the module on for test.

Unit testing was done on each of every modules that it describes under the module description :

‘1) Test for admin module:

- Testing admin for login
- Student account registration

2) Test for student login module :

- Test for student login interface
- Test for account creation

3) Test for teacher login module:

- Test for teacher login interface

#### 4.2 System of integration testing :

The primary objective is to test the module interfaces

- UI user interface module, w/c is visible to end user
- DBMS is the database management system w/c has all data
- VAL is the validation module
- CNT : these contents are displayed in reports.

#### 4.3 Performance of stress testing

Stress testing involving beyond normal operation capacity

#### 4.4 User Acceptance Testing

There are different types of acceptance testing . The most common among them is the user acceptance (UA)

#### Test Schedule

S. No	Task	Days	Start Time	End time	Responsibility
1.	Understanding and analysis	5	2 July	7 July	Team
2.	Generating test cases	10	7 July	17 July	Member 1
3.	Test case documentation	40	17 July	17 July	Member 2

4.	Verify env. step	1	17 Aug	17 Aug	Member 3
5.	Unit testing	10	18 Aug	28 Aug	Member 4
6.	IVT testing	15	7 Sept	22 Sept	Member 5
7.	Final testing	15	21 Sept	24 Sept	End user 1
8.	Eval. test criteria	2	22 Sept	24 Sept	Member 1
9.	Summary report	1	25 Sept	25 Sept	Team

### 5) Features to be tested

- GUI testing
- Database testing
- Basic operations add/delete/etc
- Advance operations
- BIU

### 6) Hardware requirements / Software requirements

- Windows -OS
- Python language
- MYSQL database



- Visual studio code - IDE

#### **7) Environment requirement**

- Mainframe- Specify both the necessary and designed properties of test acquirement
- Work Stations - computers provided in the libraries to be used by admins and students.

#### **8) Risk and mitigation**

Keep battery back up and avoid electricity issues

#### **9) Tools**

- Selenium
- 2 pp

**Result** - Successfully prepared a test plan for a library Management system

### **VIVA VOCE QUESTIONS**

#### **Q1) What is Test Planning?**

**Ans.** The activity of establishing or updating a test plan.

**Q2) What is a Test Plan?**

**Ans.** A document describing the scope, approach, resources and schedule of intended test activities. It identifies amongst others test items, the features to be tested, the testing tasks, who will do each task, degree of tester independence, the test environment, the test design techniques and entry and exit criteria to be used, and the rationale for their choice, and any risks requiring contingency planning. It is a record of the test planning process.

**Q3) What is the Output of the Test Planning phase?**

**Ans.** Test Plan Document

**Q4) Who approves the Test Plan document?**

**Ans.** Generally, the Project Manager approves the Test Plan Document after the review process.

**Q5) What is Test Point Analysis?**

**Ans.** A formula-based test estimation method based on function point analysis.

## **EXPERIMENT – 6**

**AIM:** Study the Testing Tool: Win Runner

**Theory:**

**WinRunner is a program that is responsible for the automated testing of software.**

WinRunner is a Mercury Interactive's enterprise functional testing tool for Microsoft windows applications.

Importance of Automated Testing:

1. Reduced testing time
2. Consistent test procedures – ensure process repeatability and resource independence. Eliminates errors of manual testing
3. Reduces QA cost – Upfront cost of automated testing is easily recovered over the lifetime of the product
4. Improved testing productivity – test suites can be run earlier and more often
5. Proof of adequate testing
6. For doing tedious work – test team members can focus on quality areas.

WinRunner Uses:

1. With WinRunner sophisticated automated tests can be created and run on an application.
2. A series of wizards will be provided to the user, and these wizards can create tests in an automated manner.
3. Another impressive aspect of WinRunner is the ability to record various interactions, and transform them into scripts. WinRunner is designed for testing graphical user interfaces.
4. When the user makes an interaction with the GUI, this interaction can be recorded. Recording the interactions allows to determine various bugs that need to be fixed.
5. When the test is completed, WinRunner will provide with detailed information regarding the results. It will show the errors that were found, and it will also give important information about them. The good news about these tests is that they can be reused many times.
6. WinRunner will test the computer program in a way that is very similar to normal user interactions. This is important, because it ensures a high level of accuracy and realism. Even if an engineer is not physically present, the Recover manager will troubleshoot any problems that may occur, and this will allow the tests to be completed without errors.
7. The Recover Manager is a powerful tool that can assist users with various scenarios. This is important, especially when important data needs to be recovered.

The goal of WinRunner is to make sure business processes are properly carried out. WinRunner uses TSL, or Test Script Language.

## **WinRunner Testing Modes**

### **Context Sensitive**

Context Sensitive mode records your actions on the application being tested in terms of the GUI objects you select (such as windows, lists, and buttons), while ignoring the physical location of the

object on the screen. Every time you perform an operation on the application being tested, a TSL statement describing the object selected and the action performed is generated in the test script. As you record, WinRunner writes a unique description of each selected object to a GUI map.

The GUI map consists of files maintained separately from your test scripts. If the user interface of your application changes, you have to update only the GUI map, instead of hundreds of tests. This allows you to easily reuse your Context Sensitive test scripts on future versions of your application.

To run a test, you simply play back the test script. WinRunner emulates a user by moving the mouse pointer over your application, selecting objects, and entering keyboard input. WinRunner reads the object descriptions in the GUI map and then searches in the application being tested for objects matching these descriptions. It can locate objects in a window even if their placement has changed.

## **Analog**

Analog mode records mouse clicks, keyboard input, and the exact x- and y-coordinates traveled by the mouse. When the test is run, WinRunner retraces the mouse tracks. Use Analog mode when exact mouse coordinates are important to your test, such as when testing a drawing application.

## **The WinRunner Testing Process**

Testing with WinRunner involves six main stages:

### **1. Create the GUI Map**

The first stage is to create the GUI map so WinRunner can recognize the GUI objects in the application being tested. Use the RapidTest Script wizard to review the user interface of

your application and systematically add descriptions of every GUI object to the GUI map. Alternatively, you can add descriptions of individual objects to the GUI map by clicking objects while recording a test.

### **2. Create Tests**

Next is creation of test scripts by recording, programming, or a combination of both. While recording tests, insert checkpoints where we want to check the response of the application being tested. We can insert checkpoints that check GUI objects, bitmaps, and databases. During this

process, WinRunner captures data and saves it as expected results—the expected response of the application being tested.

### **3. Debug Tests**

Run tests in Debug mode to make sure they run smoothly. One can set breakpoints, monitor variables, and control how tests are run to identify and isolate defects. Test results are saved in the debug folder, which can be discarded once debugging is finished. When WinRunner runs a test, it checks each script line for basic syntax errors, like incorrect syntax or missing elements in If, While, Switch, and For statements. We can use the Syntax Check options (Tools >Syntax Check) to check for these types of syntax errors before running your test.

### **4. Run Tests**

Tests can be run in Verify mode to test the application. Each time WinRunner encounters a checkpoint in the test script, it compares the current data of the application being tested to the expected data captured earlier. If any mismatches are found, WinRunner captures them as actual results.

### **5. View Results**

Following each test run, WinRunner displays the results in a report. The report details all the major events that occurred during the run, such as checkpoints, error messages, system messages, or user messages. If mismatches are detected at checkpoints during the test run, we can view the expected results and the actual results from the Test Results window. In cases of bitmap mismatches, one can also view a bitmap that displays only the difference between the expected and actual results.

We can view results in the standard WinRunner report view or in the Unified report view. The WinRunner report view displays the test results in a Windows-style viewer. The Unified report view displays the results in an HTML-style viewer (identical to the style used for QuickTest Professional test results).

### **6. Report Defects**

If a test run fails due to a defect in the application being tested, one can report information about the defect directly from the Test Results window.

This information is sent via e-mail to the quality assurance manager, who tracks the defect until it is fixed.

### **Using Winrunner Window**

Before you begin creating tests, you should familiarize yourself with the WinRunner main window.

#### 1.4.1 To start WinRunner:

Choose Programs -> WinRunner -> WinRunner on the Start menu.

The first time you start WinRunner, the Welcome to WinRunner window and the - What's New in WinRunner help open. From the Welcome window you can create a new test, open an existing test, or view an overview of WinRunner in your default browser.

## CASE TOOLS & SOFTWARE TESTING LAB MANUAL

If you do not want this window to appear the next time you start WinRunner, clear the Show on Startup check box. To show the Welcome to WinRunner window upon startup from within WinRunner, choose Settings -> General Options, click the Environment tab, and select the Show Welcome screen check box.

#### 1.4.2 The Main WinRunner Window

The main WinRunner window contains the following key elements:

WinRunner title bar

Menu bar, with drop-down menus of WinRunner commands

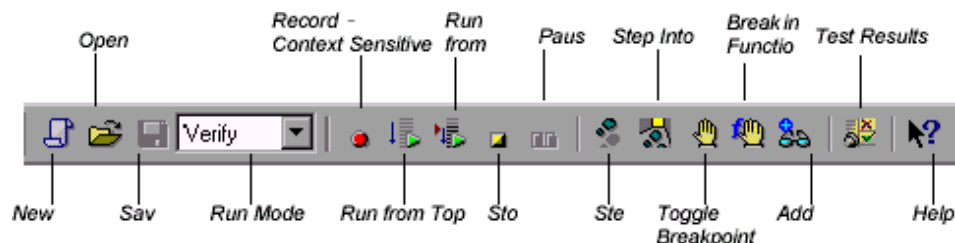
Standard toolbar, with buttons of commands commonly used when running a test User toolbar, with commands commonly used while creating a test

Status bar, with information on the current command, the line number of the insertion point and the name of the current results folder

The Standard toolbar provides easy access to frequently performed tasks, such as opening, executing, and saving tests, and viewing test results.

#### Standard Toolbar

The User toolbar displays the tools you frequently use to create test scripts. By default, the User toolbar is hidden. To display the User toolbar, choose Window > User Toolbar. When you create



tests, you can minimize the WinRunner window and work exclusively from the toolbar. The User toolbar is customizable. You choose to add or remove buttons using the Settings > Customize User Toolbar menu option. When you re-open WinRunner, the User toolbar appears as it was when you last closed it. The commands on the Standard toolbar and the User toolbar are described in detail in subsequent lessons.

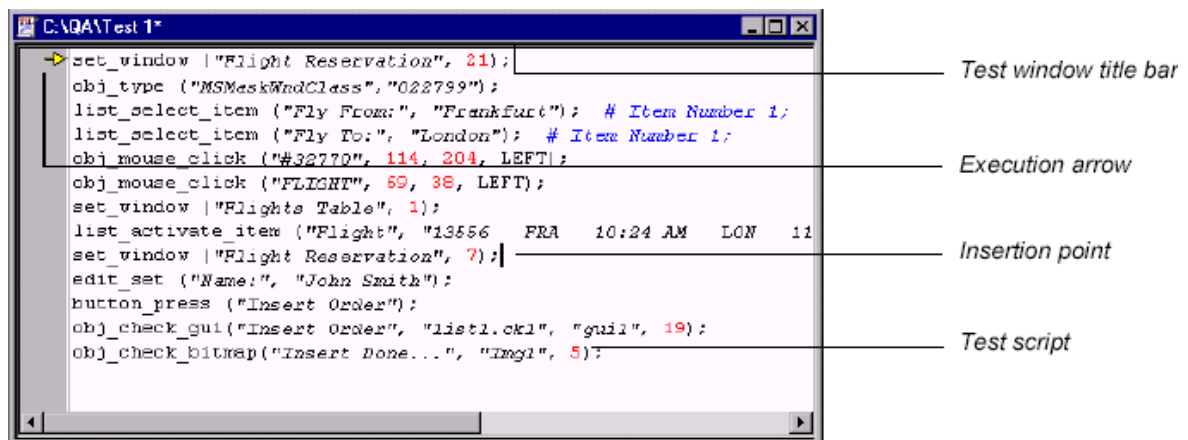
Note that you can also execute many commands using softkeys. Softkeys are keyboard shortcuts for carrying out menu commands. You can configure the softkey combinations for your keyboard using the Softkey Configuration utility in your WinRunner program group. For more information, see the - WinRunner at a Glance chapter in your WinRunner User's Guide.

Now that you are familiar with the main WinRunner window, take a few minutes to explore these window components before proceeding to the next lesson.

## The Test Window

You create and run WinRunner tests in the test window. It contains the following key elements:

- Test window title bar, with the name of the open test
- Test script, with statements generated by recording and/or programming in
- TSL, Mercury Interactive's Test Script Language
- Execution arrow, which indicates the line of the test script being executed during a test run, or the line that will next run if you select the Run from arrow option
- Insertion point, which indicates where you can insert or edit text



## VIVA VOCE QUESTIONS

**Q1. What is WinRunner?**

**Ans.** WinRunner developed by HP is a functional and automated GUI testing tool. Using this tool, users can record UI interactions and can replay them as well as test scripts. It is widely used to identify any defects in business processes.

**Q2. Specify the language used in WinRunner.**

**Ans.** WinRunner used TSL-Test Script Language. It is similar to C.

**Q3. Brief about the test scripts you've created in WinRunner.**

**Ans.** WinRunner test scripts contain statements written in Mercury Interactive's Test Script Language (TSL). We can modify the recorded test scripts, either by adding more programming elements and TSL functions or by using Function Generator, a WinRunner's visual programming tool.

**Q4. Explain WinRunner's Evaluation of Test Results.**

**Ans.** WinRunner displays test results in a report. This report lists out all the events that took place during the test run like error messages, checkpoints, user messages, or system messages. In case of any mismatch at a checkpoint, we can compare the expected and actual results in the Test Results window.

**Q5. Explain different types of modes in which we can run test scripts in WinRunner.**

**Ans.** WinRunner enables us to run test scripts in three different modes as per the phase of the testing.

- 1. Verify:** this mode helps us to check the application and compare it with earlier captured data.
- 2. Debug:** this mode helps us to find bugs in a test script.
- 3. Update:** this mode helps us to modify the expected results or generate a new expected results folder.

## **EXPERIMENT – 7**

**AIM:** Study the Test Management Tool: QA Complete.



## **THEORY:**

### **QA Complete**

- Test management tool that can be used for both manual and automated Testing
- Tool that allow us to track all aspects of software quality in an efficient manner
- Supports all aspects of test process and ensures the delivery of high quality software.

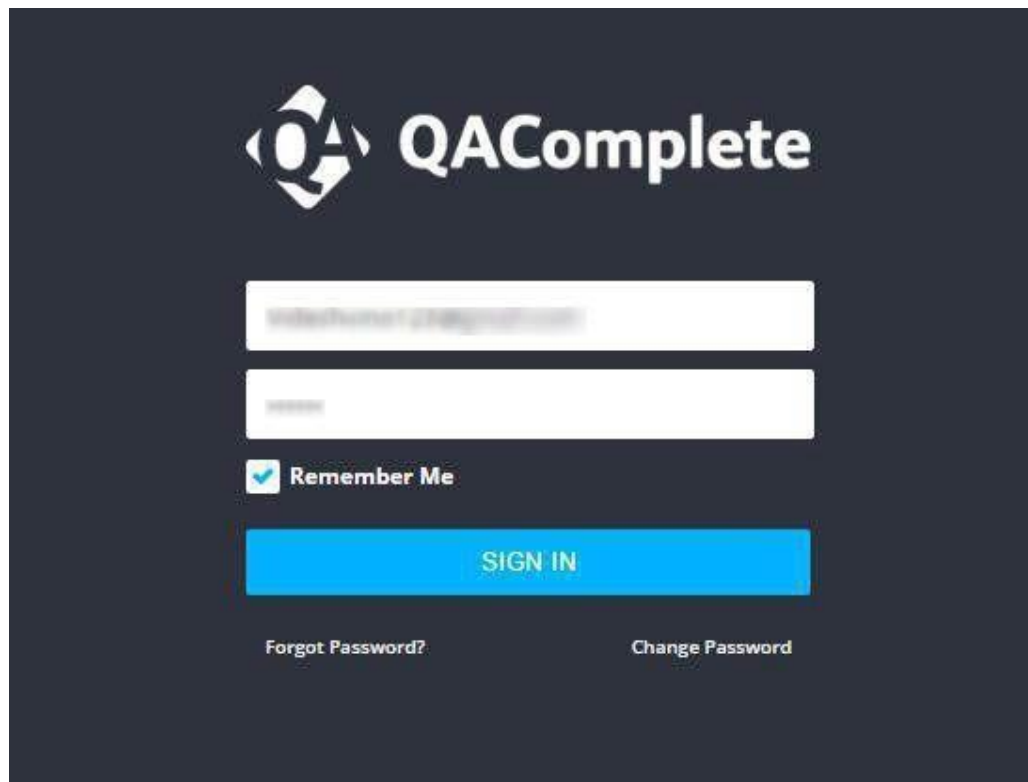
### **Benefits**

- QA Complete can be integrated with any number of tools
- Customizable as per the tester's needs
- Requirements and tests can be traced to defect effectively
- Reports and Dashboards are the key features of QA Complete

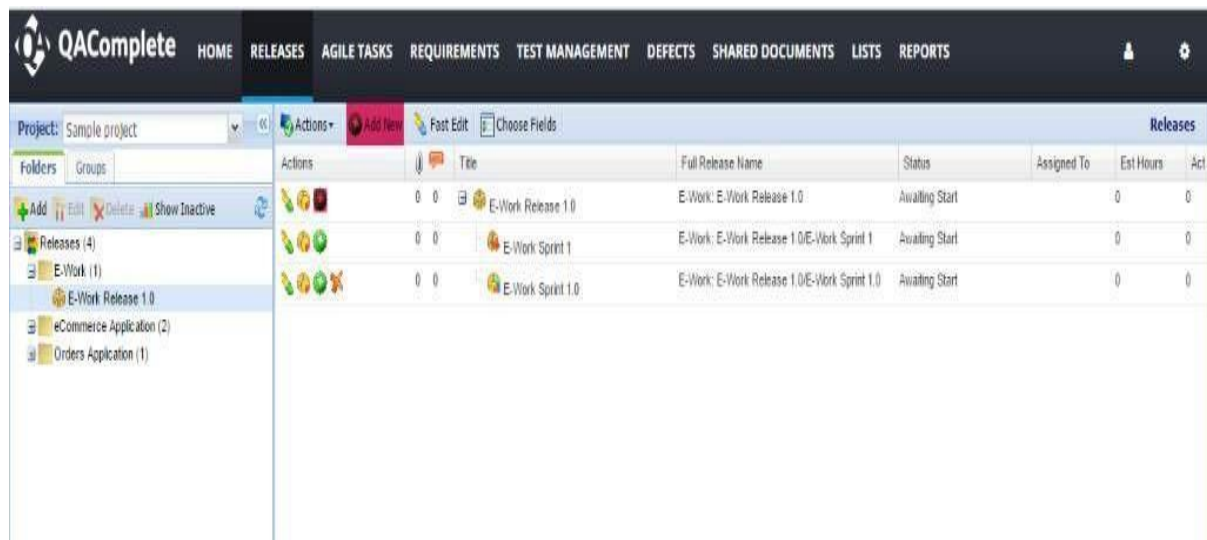
### **Features**

- Test Case Management – Simple Test Case structuring also allows for focused metrics and clear status report
- Test Environment Management – Various environments are linked to individual test cases for effective Test Coverage (across different platforms, operating systems, and devices)
- Defect and Issue Management – Mainly tracks the resolution process of bugs for each release and automatically creates bugs when test cases fail.
- Test Automation Integration – Can be integrated with various automation tools, and it allows us to track and report overall (manual and automated)
- Bug Tracker Integration – Can be integrated with various Bug tracking tools
- Shared Documents – Stores all test documents in one central location to improve collaboration.

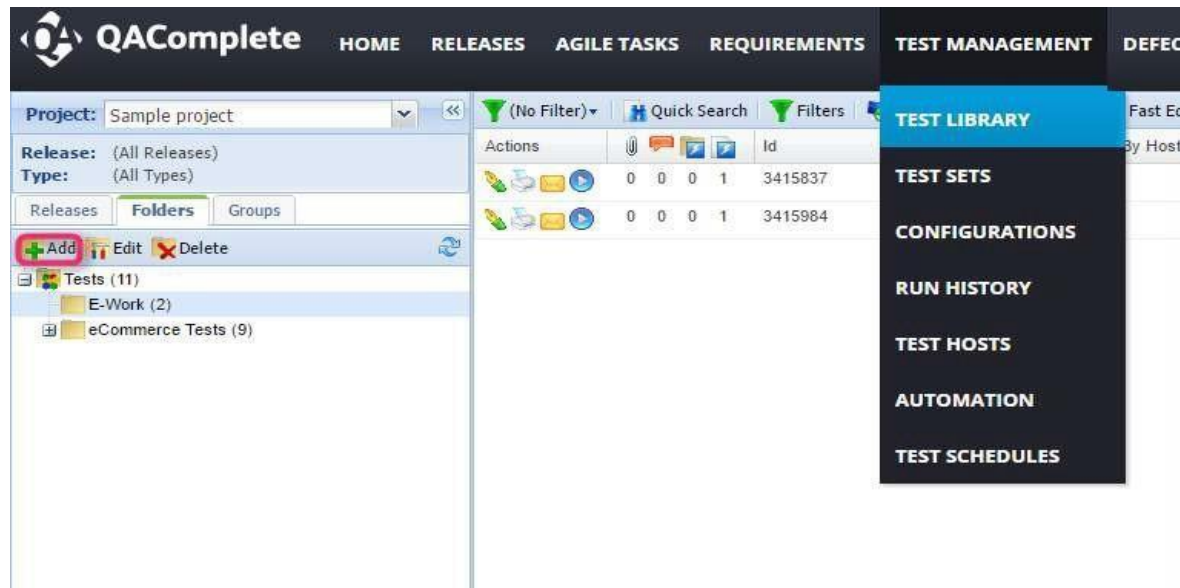
### **Step 1: Log into QA Complete Tool**



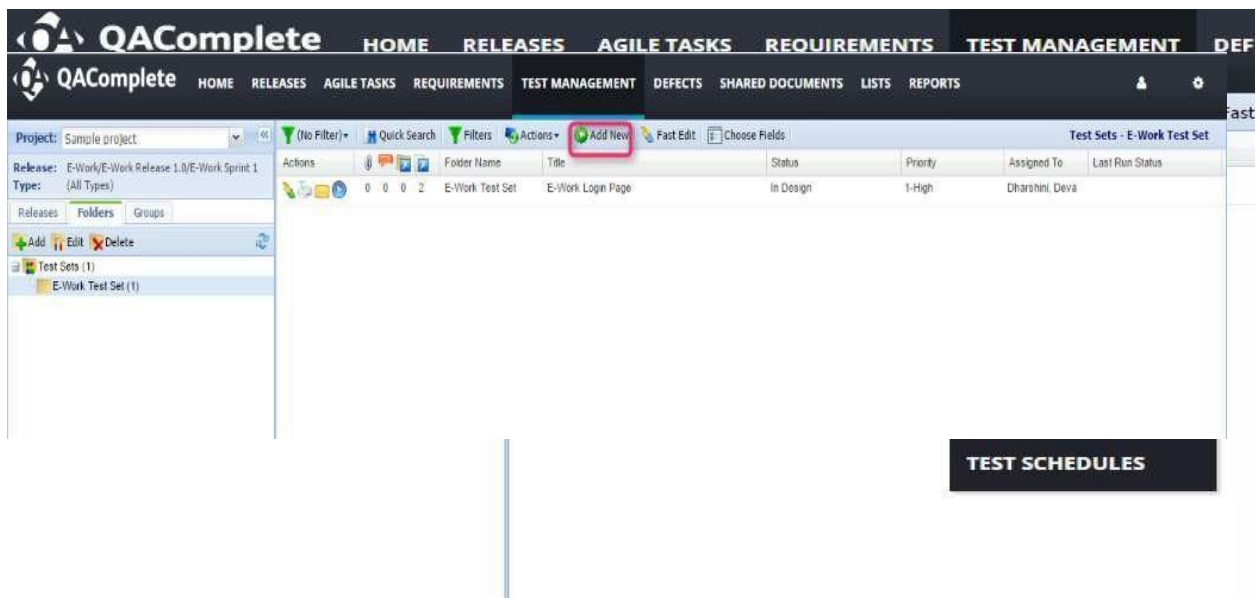
**Step 2:** Create a New Release as E-Work under the Releasetab by clicking the Add New icon and click the Add New Item to add an Iteration/Build



**Step 3:** Navigate to the Test Management Tab -> TestLibrary -> Add New folder

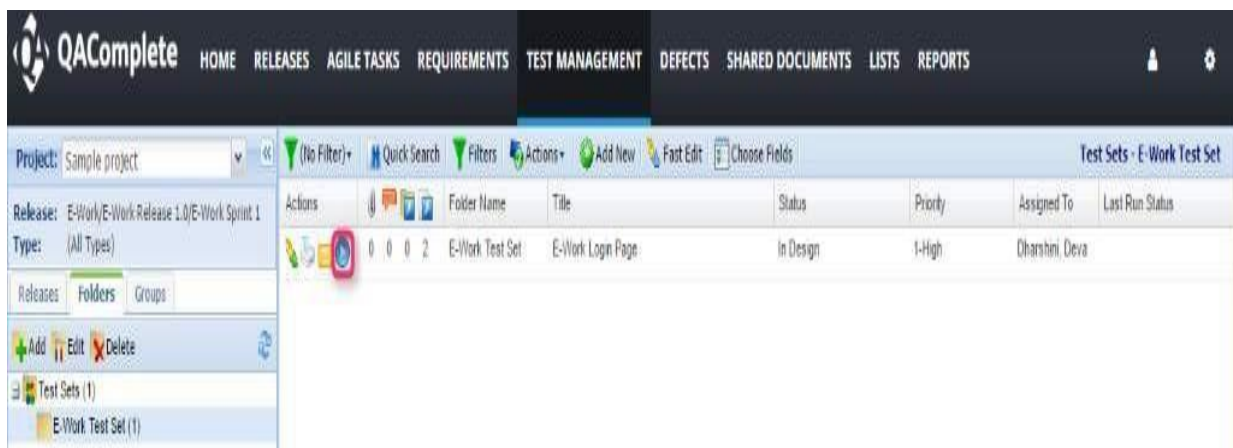


**Step 4:** Navigate to Test Management Tab -> Test Sets -> Create a folder (Add the '+' @ left panel)  
-> Create a new test set using the Add New icon -> After entering the details click the Submit button  
-> Navigate to the Tests Tab and Design the steps accordingly

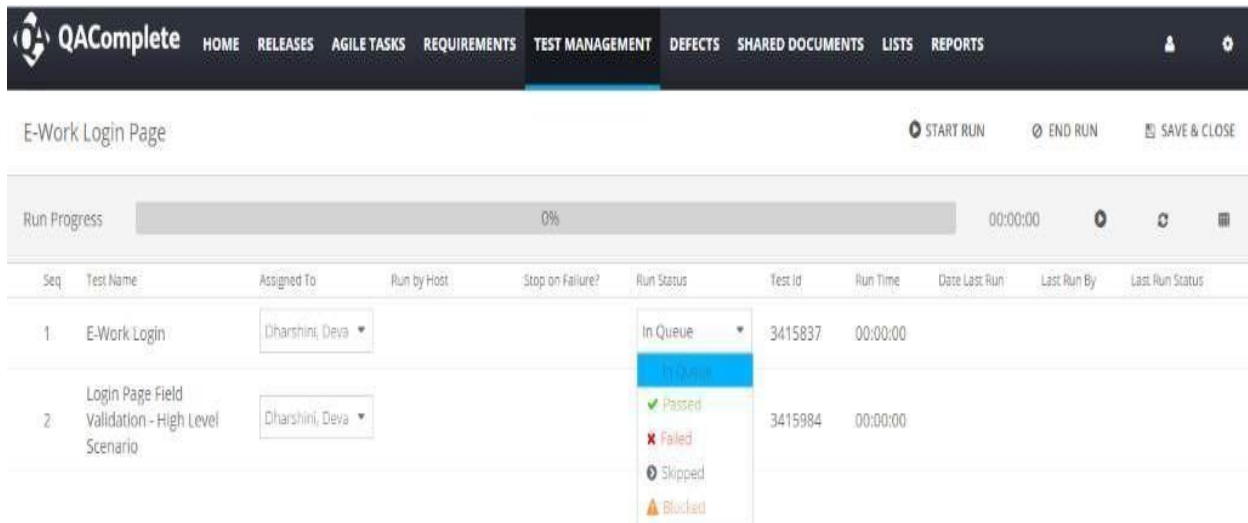




**Step 5:** To run the Test Sets -> Test Management Tool -> Test Sets->



**Step 6:** Click the Start Run at the top right corner. Based on the working functionality, select the Run Status and click the End Run option finally.



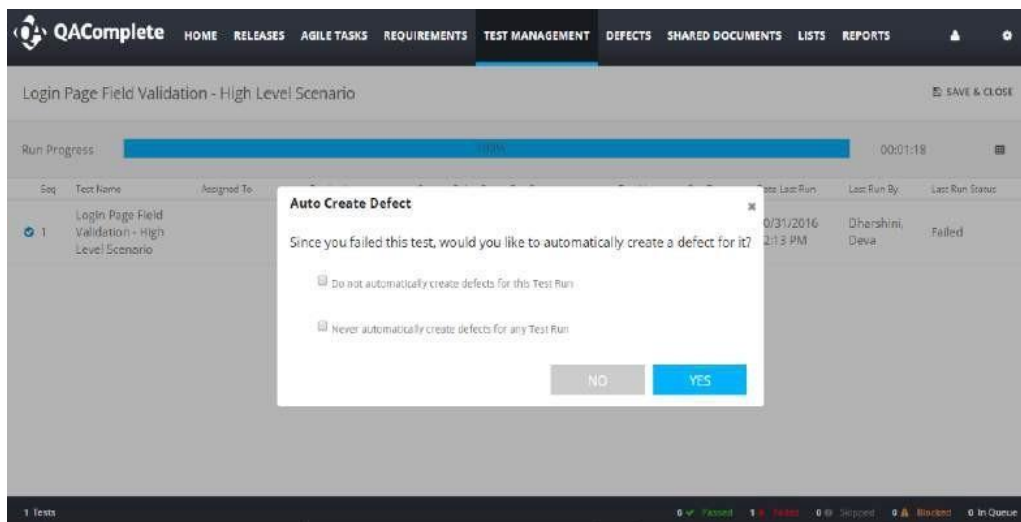
QAComplete HOME RELEASES AGILE TASKS REQUIREMENTS TEST MANAGEMENT DEFECTS SHARED DOCUMENTS LISTS REPORTS

E-Work Login Page START RUN END RUN SAVE & CLOSE

Run Progress 0% 00:00:00

Seq	Test Name	Assigned To	Run by Host	Stop on Failure?	Run Status	Test Id	Run Time	Date Last Run	Last Run By	Last Run Status
1	E-Work Login	Dharshini, Deva			In Queue	3415837	00:00:00			
2	Login Page Field Validation - High Level Scenario	Dharshini, Deva			In Queue	3415984	00:00:00			

**Step 7:** If any of the steps fail in a Test Set during the run, it prompts to create a defect automatically.



QAComplete HOME RELEASES AGILE TASKS REQUIREMENTS TEST MANAGEMENT DEFECTS SHARED DOCUMENTS LISTS REPORTS

Login Page Field Validation - High Level Scenario SAVE & CLOSE

Run Progress 100% 00:01:18

**Auto Create Defect**

Since you failed this test, would you like to automatically create a defect for it?

☒ Do not automatically create defects for this Test Run

☐ Never automatically create defects for any Test Run

NO YES

Seq	Test Name	Assigned To	Run by Host	Stop on Failure?	Run Status	Test Id	Run Time	Date Last Run	Last Run By	Last Run Status
1	Login Page Field Validation - High Level Scenario	Dharshini, Deva			Failed	3415984	00:01:18	10/31/2016 2:13 PM	Dharshini, Deva	Failed

1 Tests 0 Passed 1 Failed 0 Skipped 0 Blocked 0 In Queue

**Step 8:** When the YES option is selected,a defect is created automatically

Edge UI Listing - Google Chrome

https://app.qaccomplete.smartbear.com/common/Logon.asp?WCI=tplQuickEdit2&FormName=Bugs&ID=0&TestRunItemId=3822549&ConfigId=0&Relea

### Add Defect

**Auto-fill from Recent Entry:**

**Folder Name:** Select a folder **Id:** (Auto Generated)

**\* Title:** Login Page Field Validation - High Level Scenario **\* Status:** Active

**Priority:** **Owner:** Dharshini, Deva

**Assigned To:**

**Description:** Tag Arial B I U A\* A\* As this functionality is in the Design Phase Test Case has been written in a high level




**Steps to Reproduce:** Step #1(Failed)  
Step: 1. Enter the value in the UserName field User Name - 613772 2. Enter the value in the password field Password - abc@d2 3. Click on the Submit button Note : - UserName should be only Numeric characters - Password should be alpha numeric  
Expected Result: Should navigate to the E- Work home page

Cancel Submit Submit/Add another

**Step 9:** Navigate to the Defects Tab and viewthe automatically created bug(s).

QAComplete HOME RELEASES AGILE TASKS REQUIREMENTS TEST MANAGEMENT DEFECTS SHARED DOCUMENTS LISTS REPORTS

Project: Sample project (No Filter) Quick Search Filters Actions Add New Fast Edit Choose Fields Defects

Actions	Id	Title	Status	Severity	Priority	Assigned To	Date Open...
	0 0 3542592	Log in form allowed invalid password.	Active	2-Major Bug	1-Fix ASAP		07/29/2015
	0 0 3542593	New user form did not prevent user name less L.	Active	2-Major Bug	1-Fix ASAP		07/29/2015
	0 0 3543512	Login Page Field Validation - High Level Scenario	Active				10/31/2016

Recent Items

# TEST CASES GENERATION

The screenshot displays the QAComplete SmartBear web application interface. The top navigation bar includes links for HOME, RELEASES, AGILE TASKS, REQUIREMENTS, TEST MANAGEMENT, DEFECTS, SHARED DOCUMENTS, LISTS, and REPORTS. The main content area shows a list of defects under the 'DEFECTS' tab. The defects are organized into columns: ID, Title, Status, Severity, Priority, Assigned To, Date Opened, Functional Area, Default WBS, and Folder Name. The list includes various defects such as 'Email not sent upon Defect creation', 'Handwritten sample!!!!', 'Log in form showed invalid password!!', 'New user form did not prevent user name less t', 'Add new user with valid name and details!!!!', 'Existing user logs in with valid email, invalid pas', 'Add new user with valid name and details', 'Close/Login in been allowed invalid password', 'sample defect', 'DEMO SAMPLE', 'sample v42 defect', 'sample CA defect', 'Invalid new defect', 'Invalid Sample', 'v42 sample defect', 'Close/DEMO SAMPLE', 'Close/Invalid new defect', 'Close/sample CA defect', 'Close/sample v42 defect', 'Close/Invalid Sample', 'sample', and 'Add new user with valid name and details!!!!'. The interface also features a sidebar on the right with a 'Filter' section and a 'Choose Fields' section. The bottom of the page shows a 'Page: 1 of 1' indicator and a 'Close' button.

ID	Title	Status	Severity	Priority	Assigned To	Date Opened	Functional Area	Default WBS	Folder Name
3871219	Email not sent upon Defect creation	Open	2-Fix Soon	2-Fix ASAP	Paragang, Khat	09/02/2017			UP
3816070	Handwritten sample!!!!	Open		1-Fix ASAP	Paragang, Khat	11/10/2017			
3462263	Log in form showed invalid password!!	Accepted	2-Major Bug	1-Fix ASAP	Paragang, Khat	07/29/2015			HRB
3462264	New user form did not prevent user name less t	Accepted	4-Triple Bug	1-Fix ASAP	Paragang, Khat	07/29/2015			HRB
3507771	Add new user with valid name and details!!!!	Accepted	2-Major Bug		Paragang, Khat	06/01/2016			HRB
3506981	Existing user logs in with valid email, invalid pas	WIP	2-Major Bug		Paragang, Khat	02/27/2017			UP
3506104	Add new user with valid name and details	WIP	4-Triple Bug		Paragang, Khat	02/29/2017			HRB
3507771	Close/Login in been allowed invalid password	Rejected	4-Triple Bug	1-Fix ASAP	Paragang, Khat	07/29/2015			HRB
3743954	sample defect	WIP	3-Workaround			05/02/2017			
3743956	DEMO SAMPLE	Deferred	1-Crash	1-Fix ASAP	Chopra, Pawan	05/04/2017			
3753284	sample v42 defect	Open	3-Workaround			05/16/2017			Duplicates
3753286	sample CA defect	WIP	3-Workaround			05/16/2017			Duplicates
3756717	Invalid new defect	Rejected	1-Crash			05/17/2017			Duplicates
3759432	Invalid Sample	Open	1-Crash			05/02/2017			Duplicates
3759433	v42 sample defect	WIP	1-Crash			05/02/2017			Duplicates
3759521	Close/DEMO SAMPLE	Deferred	4-Triple Bug	2-Fix Soon	Chopra, Pawan	05/04/2017			Duplicates
3759522	Close/Invalid new defect	Open	4-Triple Bug			05/17/2017			Duplicates
3759523	Close/sample CA defect	Rejected	3-Workaround			05/16/2017			Duplicates
3759524	Close/sample v42 defect	Open	4-Triple Bug			05/16/2017			Duplicates
3759525	Close/Invalid Sample	Open	4-Triple Bug			06/02/2017			Duplicates
3762236	sample	Fixed				05/02/2017			
3773504	Add new user with valid name and details!!!!	Accepted				06/17/2017			

## **VIVA VOCE QUESTIONS**

### **Q1) What is data - driven automation?**

**Ans.** Testing the functionality with more test cases becomes laborious as the functionality grows. For multiple sets of data (test cases), you can execute the test once in which you can figure out for which data it has failed and for which data, the test has passed.

### **Q2) How you will evaluate the tool for test automation?**

**Ans.** We need to concentrate on the features of the tools and how this could be beneficial for our project. The additional new features and the enhancements of the features will also help.

### **Q3) What are main benefits of test automation?**

**Ans.** The main benefits of test automation are that it is fast, reliable, comprehensive, reusable

### **Q4) What testing activities you may want to automate?**

**Ans.** Automate all the high priority test cases which needs to be executed as a part of regression testing for each build cycle.

### **Q5) With how many tools can QA complete be integrated ?**

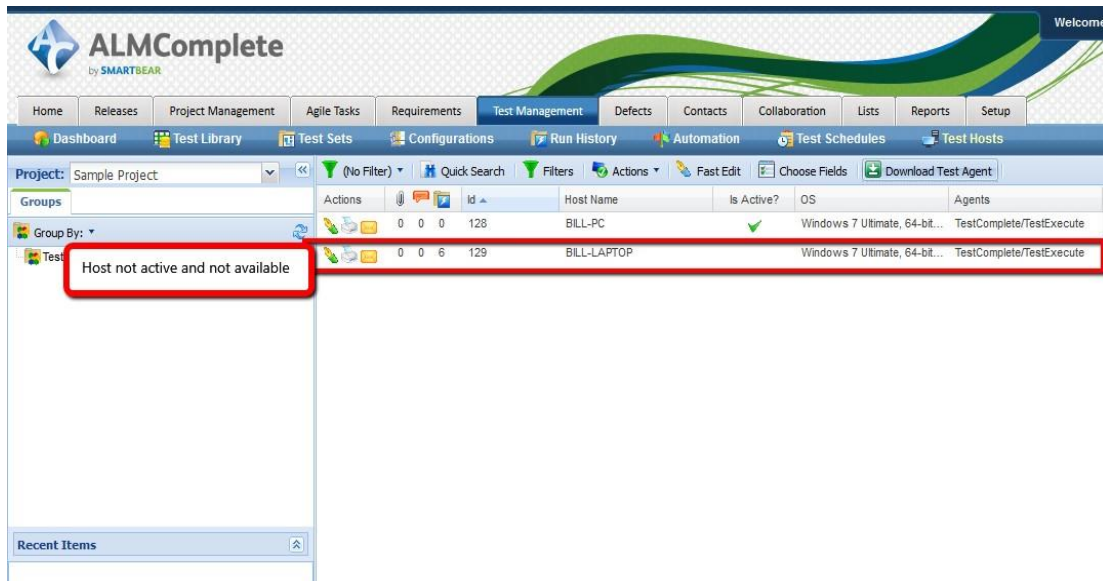
**Ans.** A Complete can be integrated with any number of tools



## **EXPERIMENT – 8**

**AIM:** Automate the Test cases using Test Automation tool QA Complete.

### **Setting up automated tests in QA complete**



#### **Step1:**

Check in QAComplete that the TestComplete host is available by viewing the 'Test Hosts' records. if the host isn't listed at all then enable the 'Show Inactive Test Hosts' option. If the host isn't active then start the service on the TestComplete machine.

**Step 2:** On the TestComplete machine Press Ctrl+Shift+Esc to display task manager and then click on the 'Services' tab followed by the 'Services' button. In the Services window click on the 'TestManager Agent' service and start the service.

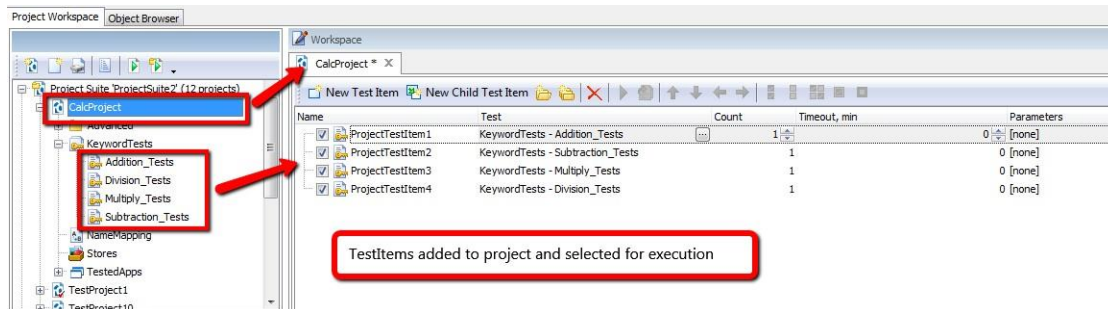
#### **Step 3: Creating An Automate Test**

This is a 4 stage process.

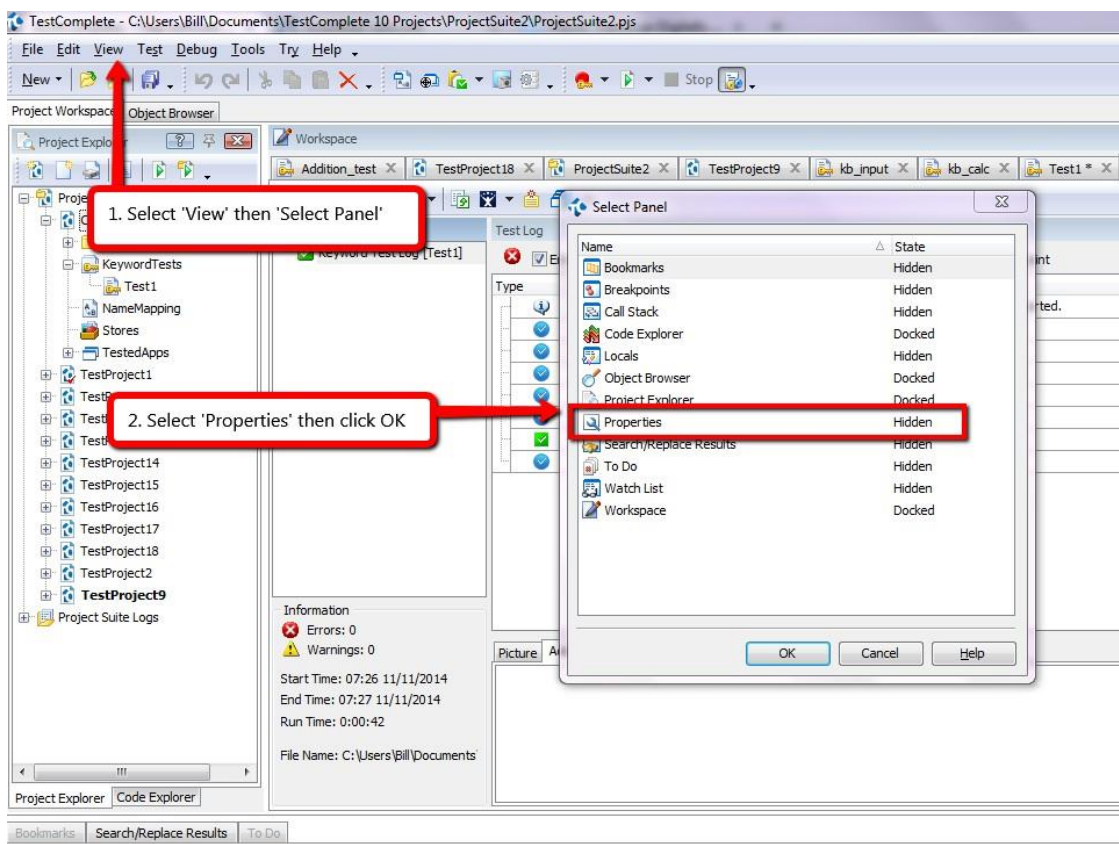
1. Package up the TestComplete project suite
2. Define the Automated Test in the QAComplete Test Library
3. Execution of Automated Tests – standalone
4. Execution of Automated Tests – as part of a Test Set

1. Package up the TestComplete Project Suite To zip the project suite up follow these steps:

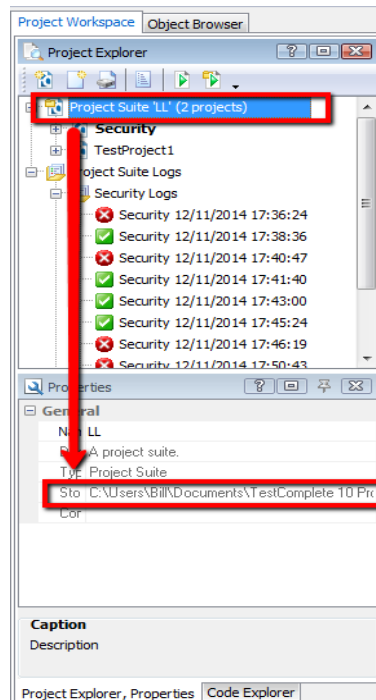
1) Make sure to define the 'Test Items' and enabled them within TestComplete project(s)



2) Find location of testcomplete project suite on the file system of test complete machine



From here we can see where test complete is storing the project on file system.



3) On the file system (or in testcomplete) remove the log files.

4) At the project suite level on the file system find the folder containing project suite and zip up this project suite.

## 2. Define the Automated Test in the QAComplete Test Library:

Create the test case in the 'Test Library' area of QAComplete and then attach the zipped up TestComplete project suite to this test case.

5) First we need to create a new test. Navigate to the Test Management Library area in QAComplete and select 'Add New'. Then we need to define the usual meta data required to create the test case (e.g. Title, Description, etc). A couple of fields that are important though:

- Execution Type: set this to Automated
- Default Host Name: set this to the host that will be used by default to execute the automated test

Assuming selected Execution Type = Automated then save the test case to the 'Automations' tab for the test case. Click 'Add New' to add a new TestComplete Project Suite.

When adding a new TestComplete project suite to QAComplete following 6 fields will be presented:

- **Title:** either leave this blank and QAComplete will give this automated test the same name as the TestComplete project or define your own name
- **Time Out:** this is how long it should take to run the test. If it goes past this time out value then the test runner will stop running the test and move on to the next one.
- **Entry Point:** use this to identify a specific test or project to run. If this field is blank the whole project suite will be run. Specify a specific test case to execute an individual test case or a specific project to run only one project.
- **Agent:** at the moment QAComplete only supports one type of test agent which is TestComplete/TestExecute. Other types of test agent are in the pipeline.
- **Web Site Address or UNC Path:** place the zipped up project suite file on a shared drive. In which case, define the path to that location and the file name here. Alternatively...
- **File Attachments:** attached the zipped up project suite file to the QAComplete test case and upload the file to QAComplete

A completed record with a project suite zip file uploaded looks like this (the entry point in this example is at the Test Case level)

A completed record with a UNC path looks like this (The Entry point in this example is at the project level)...

**Add Automation**

[Actions](#) [Help](#)

[Return To Listing](#)

Title:	CalcProject
Time Out(sec.):	30
Entry Point:	CalcProject
Created By:	11/14/2014 8:09:31 AM by Echlin, Bill
* Agent:	TestComplete/TestExecute
Last Updated:	
Web Site Address or UNC Path:	file:///B:\\BILL-LAPTOP\\Users\\Bill\\Documents\\TestComplete 10 Projects\\ProjectSuite1.zip
File Attachments:	<input type="text"/> <input type="button" value="Reset"/> <input data-bbox="1377 1331 1442 1352" type="button" value="Browse..."/>

At this point in time, only add one automation project suite to a single QAComplete automated test case

## **VIVA VOCE QUESTIONS**

### **Q1) What are the most popular tools for automation testing?**

**Ans.** The most popular test tool for automation testing are:

Selenium

UFT.

Rational Robot.

Here is a complete list of automation testing tools.

### **Q2) How can you measure the success of automation testing?**

**Ans.** Following criteria can map the success of automation testing:

Defect Detection Ratio

Automation execution time and time savings to release the product

Reduction in Labour & other costs

### **Q3) Can you list out some disadvantages of manual testing?**

**Ans.** Manual testing requires more time and more resources.

Inaccuracy

Executing the same test case repeatedly is error-prone and tedious.

It is impractical to do manual testing on very large and time-bound projects.

### **Q4) What are the differences between open-source tools, vendor tools, & in-house tools in automation testing?**

**Ans.** Here are the differences between all:

Open-Source Tools: They are free tools with source code available on the internet. Example: Selenium

Vendor Tools: These testing tools are developed by companies, and you need to purchase their licenses. Example: Microfocus UFT.

In-house Tools: It is built by companies for their use.

## **EXPERIMENT - 9**

**AIM:** Learn how to raise and report Bugs using Bug tracking tool (Bugzilla, Jira using QA Complete)

### **Introduction to Bugzilla**

Bugzilla is an open-source issue/bug tracking system that allows developers effectively to keep track of outstanding problems with their product. It is written in perl and uses mysql database.

Bugzilla is a defect tracking tool, however it can be used as a test management tool as such it can be easily linked with other test case management tools like quality center, testlink etc.

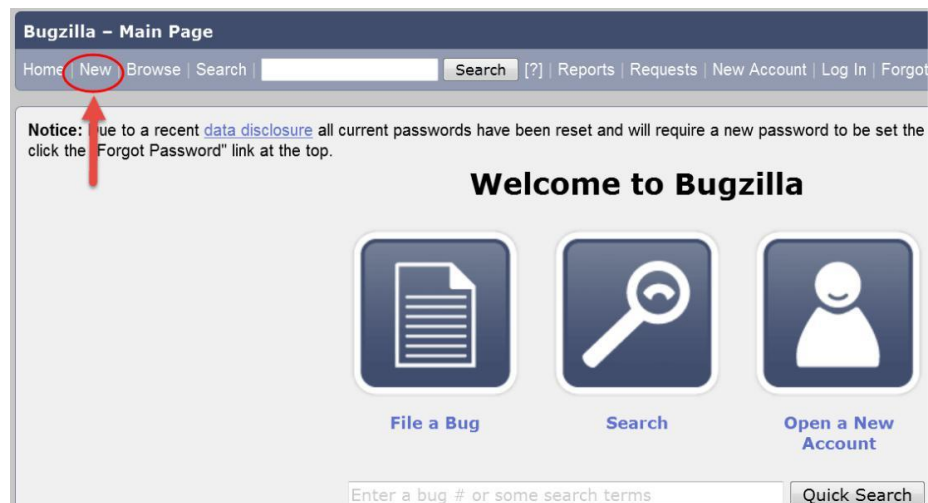
This open bug-tracker enables users to stay connected with their clients or employees, to communicate about problems effectively throughout the data-management chain.

Key features of bugzilla includes:

- Advanced search capabilities
- E-mail notifications
- Modify/file bugs by e-mail
- Time tracking
- Strong security
- Customization
- Localization

### **Creating a bug-report in bugzilla**

Step 1) To create a new bug in bugzilla, visit the home-page of bugzilla and click on new tab from the main menu



**Step 2)** In the next window

1. Enter Product
2. Enter Component
3. Give Component description
4. Select version,
5. Select severity
6. Select Hardware
7. Select OS
8. Enter Summary
9. Enter Description
10. Attach Attachment
11. Submit

The screenshot shows a web browser window with a bug reporting form. The form is titled "Bug Report" and includes a navigation bar with links: Home, New, Browse, Search, Reports, My Requests, Preferences, Help, and Log Out. A notice at the top states: "Notice: Due to a recent data disclosure all current passwords have been reset and will require a new password to be set. Before reporting a bug, please read the bug writing guidelines, please look at the list of most frequently reported bugs." The form fields are numbered 1 through 11: 1. Product: Sam's Widget; 2. Component: Widget Gears; 3. Component Description: Gears for Sam's widgets; 4. Version: unspecified; 5. Severity: normal; 6. Hardware: PC; 7. OS: Windows NT; 8. Summary: (empty); 9. Description: (empty); 10. Attachment: Add an attachment; 11. Submit Bug. A green message at the bottom right says: "We've made a guess at your operating system and platform, check them and make any corrections if necessary."

Step 4) Bug is created ID# 26320 is assigned to our Bug. We can also add additional information to the assigned bug like URL, keywords, whiteboard, tags, etc. This extra-information is helpful to give more detail about the Bug we have created.

1. Large text box
2. URL
3. Whiteboard
4. Keywords

5. Tags
6. Depends on
7. Blocks
8. Attachments

First Last Prev Next This bug is not in your last search results.

**Bug 26320 - Gears for sams widget twisted (edit)**

Status: **CONFIRMED** (edit)

Product: Sam's Widget  
 Component: Widget Gears  
 Version: unspecified  
 Hardware: PC Windows NT

Importance: P2 - normal  
 Milestone: ---  
 Assigned To: sam.folk.williams (edit) (take)  
 QA Contact: (edit) (take)

2 URL:  
 3 Whiteboard:  
 4 Keywords:  
 5 Tags:  
 6 Depends on:  
 7 Blocks:

Show dependency tree / graph

Reported: 2015-01-07 02:50 PST by James  
 Modified: 2015-01-07 03:10 PST (history)  
 CC List: 1 user including you (edit)

See Also: (add)  
 Large text box: 1

A multiple-select box: Always Appears, Also Always Appears, Third Value, Always

Drop Down List:  
 Date Time:  
 Bug ID Field:

Flags: None yet set (set flags)

Orig. Est.:	Current Est.:	Hours Worked:	Hours Left:	%Complete:	Gain:	Deadline:
0.0	0.0	0.0 + 0	0.0	0	0.0	2015-01-09

Summarize time (including time for bugs blocking this bug)

Attachments  
 Add an attachment (proposed patch, testcase, etc.)

8

Step 5) in the same window if scroll down further then select deadline date and also status of the bug. Deadline in bugzilla usually gives the time-limit to resolve the bug in given time frame

Orig. Est.:	Current Est.:	Hours Worked:	Hours Left:	%Complete:	Gain:	Deadline:
0.0	0.0	0.0 + 0	0.0	0	0.0	

Summarize time (including time for bugs blocking this bug)

Attachments  
 Add an attachment (proposed patch, testcase, etc.)

Additional Comments:

Status: **CONFIRMED**  
 CONFIRMED  
 IN\_PROGRESS  
 RESOLVED

James 2015-01-07 02:50:31 PST

Description [reply] [-]

The widget gears are twisted at the end and not showing correct signal

2015 January 2015  
 Su Mo Tu We Th Fr Sa  
 28 29 30 31 1 2 3  
 4 5 6 7 8 9 10  
 11 12 13 14 15 16 17  
 18 19 20 21 22 23 24  
 25 26 27 28 29 30 31  
 1 2 3 4 5 6 7

Save Changes

Collapse All Comments  
 Expand All Comments



## Create graphical reports

In order to represent that in the graph, select severity on x-axis and component on y-axis, and then click on generate report. It will generate a report with crucial information.

**Bugzilla - Generate Graphical Report**

Home | New | Browse | Search | [?] | Reports | My Requests | Preferences | Help | Log out jeegarguru@gmail.com

**Notice:** Due to a recent [data disclosure](#) all current passwords have been reset and will require a new password to be set the next time this site is accessed. To do so, click the "Forgot Password" link at the top.

Choose one or more fields as your axes, and then refine your set of bugs using the rest of the form.

1 **Vertical Axis:** Severity

2 **Plot Data Sets:** ☒ Individually ☐ Stacked

3 **Horizontal Axis:** Component

4 **Multiple Images:** <none>

5 **Format:** ☐ Line Graph ☒ Bar Chart ☐ Pie Chart

6 **Summary:** contains all of the strings

12 **Generate Report**

7 **Classification:** Unclassified Widgets Mercury

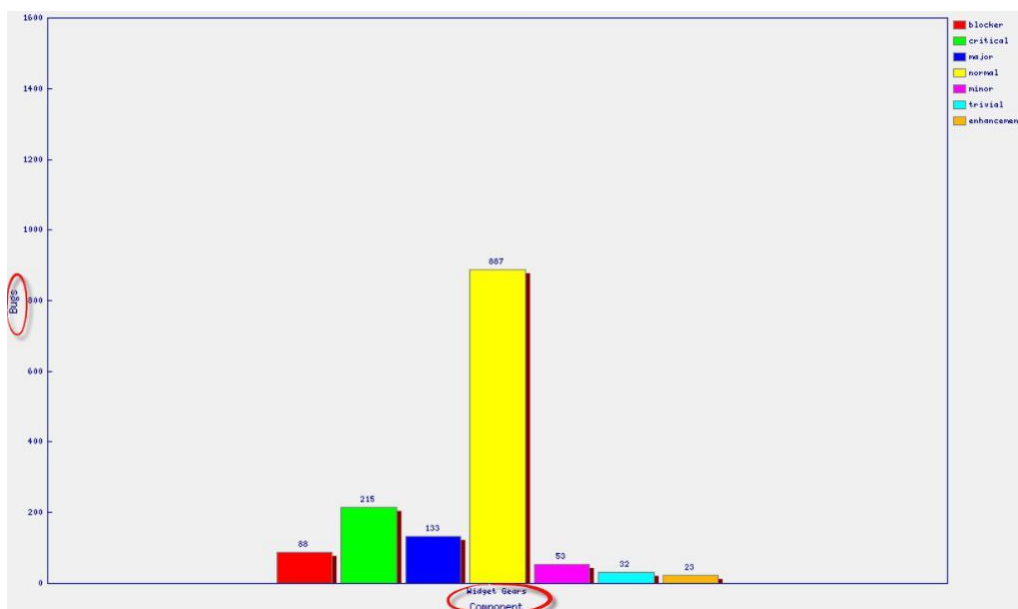
8 **Product:** Sam's Widget

9 **Component:** Widget Gears

10 **Status:** UNCONFIRMED CONFIRMED IN\_PROGRESS RESOLVED VERIFIED

11 **Resolution:** --- FIXED INVALID WONTFIX LATER REMIND DUPLICATE

The graph below shows the bar chart representation for the bugs severity in component "widget gears". In the graph below, the most severe bug or blockers in components are 88 while bugs with normal severity are at top with 667 numbers.



## **VIVA VOCE QUESTIONS**

### **Q1) What is the deployment environment for Bugzilla**

**Ans.** Bugzilla developed in LAMP (Linux, Apache, MySQL, and PHP) technology.

### **Q2) What is the difference between Quality Center and Bugzilla?**

**Ans.** Quality Center is a Test Management Tool, supports Complete Application Life Cycle Management, Requirements Management, Test Design, Test Execution, Defect Management, Traceability.

Bugzilla supports Defects Management only.

### **Q3) What are the advantages of Bugzilla?**

**Ans.** Bugzilla can increase the productivity and accountability of individual employees by providing a documented workflow and positive feedback for good performance.

### **Q4) How to create a User Account in Bugzilla?**

**Ans.** a. Enter “E-mail address” and “Real Name” in the spaces provided, then select the “Create Account” button.

b. Within moments, you should receive an email to the address you provided above, which contains your login name (generally the same as the email address), and a password that you can use to access your account.

### **Q5) What is The Bugzilla Query Page?**

**Ans.** The Bugzilla Query Page is the heart and soul of the Bugzilla user experience. It is the master interface where we can find any bug report, comment, or patch currently in the Bugzilla system.

## **EXPERIMENT - 10**

**AIM:** Study of any open source testing tool (Web Performance Analyzer/OSTA)

### **Theory:**

OpenSTA is a [feature-rich](#) GUI-based [web server benchmarking](#) utility that can perform scripted HTTP and HTTPS heavy load tests with performance measurements.<sup>[1]</sup> It is freely available and distributable under the [open source GNU General Public License](#). OpenSTA currently only runs on [Microsoft Windows](#)-based operating systems.

Scripts are recorded in a [proprietary language](#) called "SCL". It is a fairly simple coding language that provides support for custom functions, variable scopes, and random or sequential lists.

OpenSTA was originally written by Cyrano. Cyrano's intentions were to write commercial plug in modules and support for OpenSTA for performance testing non-web applications.

### **Web Load Testing**

#### ***HTTP Stress & Performance Tests***

The applications that make up the current OpenSTA toolset were designed to be used by performance testing consultants or other technically proficient individuals. This means testing is performed using the record and replay metaphor common in most other similar commercially available toolsets. Recordings are made in the tester's own browser producing simple scripts that can be edited and controlled with a special high level scripting language. These scripted sessions can then be played back to simulate many users by a high performance load generation engine. Using this methodology a user can generate realistic heavy loads simulating the activity of hundreds to thousands of virtual users.

### **Data Collection**

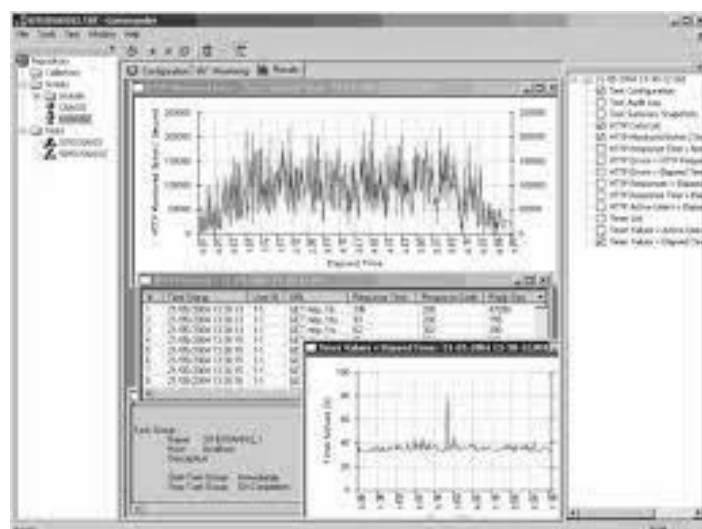
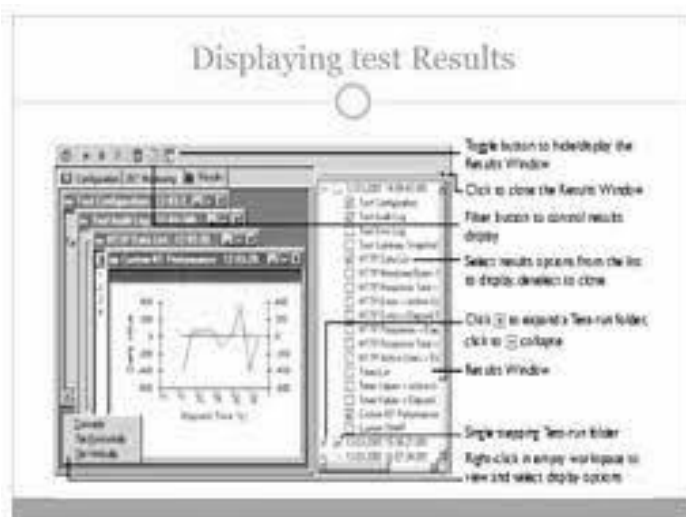
#### ***Timers, Windows Performance & SNMP Statistics***

Results and statistics are collected during test runs by a variety of automatic and user controlled mechanisms. These can include scripted timers, SNMP data, Windows Performance Monitor

stats and HTTP results & timings. Much of the data logged can be monitored live during the test runs; once test runs are complete, logs can be viewed, graphed, filtered and exported for use by more sophisticated report generation software.

### ***Completely free & open source***

The OpenSTA toolset is Open Source software licensed under the GNU GPL (General Public License), this means it is free and will always remain free. If you wish to build your own customized version of OpenSTA or take part in its development then you can complete toolset source code, buildable in Microsoft Visual Studio 6.



## **VIVA VOCE QUESTIONS**

### **Q1) What is the need for doing a performance test?**

**Ans)** In order to evaluate performances that are applications based we need performance testing. These evaluations are performed under certain stress and load conditions. The response time that is related to the activity of users is generally used for measuring [performance testing](#). The entire system is tested at a high condition of stress and load.

### **Q2) Mention the various types of performance testing?**

**Ans)** There are three types of performance testing which are listed below.

1. Load performance testing: This type of testing is utilized when the user wants to analog to testing that is volume-related. And it is also used for determining how to do application work with a huge amount of data.
2. Stress performance testing: Applications are examined on the basis of their behavior during peak activity bursts.
3. Capacity performance testing: The entire capacity of the system is measured and also it helps in detailing at what time the response time will become unacceptable.

### **Q3) What do you mean by concurrent user hits in case of load test?**

**Ans)** There may arise a situation when more than one user will hit on the same event of the application during load testing and this situation is referred to as concurrent user hit. There is the addition of these concurrency points. This is done so that the virtual multiple users would be able to work in a single event of application.

### **Q4) How is it possible to identify situations related to performance bottlenecks?**

**Ans)** Through the process of monitoring applications that go against the stress and load conditions, bottlenecks can be identified. Also, there is the requirement of load runner. Load runner is required since various kinds of monitors are provided by it like that of the monitor of the web resource, firewall monitor, runtime monitor, database server, and network delay.

### **Q5) Name the phases of the performance test?**

**Ans)** The phases of performance test:

- Design or planning,
- Building
- Execution and
- Tuning and analyzing