LAB MANUAL OF

SOFTWARE TESTING AND QUALITY ASSURANCE

ETCS -453

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

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

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

VISION

To nurture young minds in a learning environment of high academic value and imbibe spiritual and ethical values with technological and management competence.

MISSION

The Institute shall endeavor to incorporate the following basic missions in the teaching methodology:

Engineering Hardware – Software Symbiosis

Practical exercises in all Engineering and Management disciplines shall be carried out by Hardware equipment as well as the related software enabling deeper understanding of basic concepts and encouraging inquisitive nature.

Life – Long Learning

The Institute strives to match technological advancements and encourage students to keep updating their knowledge for enhancing their skills and inculcating their habit of continuous learning.

Liberalization and Globalization

The Institute endeavors to enhance technical and management skills of students so that they are intellectually capable and competent professionals with Industrial Aptitude to face the challenges of globalization.

Diversification

The Engineering, Technology and Management disciplines have diverse fields of studies with different attributes. The aim is to create a synergy of the above attributes by encouraging analytical thinking.

Digitization of Learning Processes

The Institute provides seamless opportunities for innovative learning in all Engineering and Management disciplines through digitization of learning processes using analysis, synthesis, simulation, graphics, tutorials and related tools to create a platform for multi-disciplinary approach.

Entrepreneurship

The Institute strives to develop potential Engineers and Managers by enhancing their skills and research capabilities so that they become successful entrepreneurs and responsible citizens.

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE & ENGINEERING DEPARTMENT

VISION

To Produce "Critical thinkers of Innovative Technology"

MISSION

To provide an excellent learning environment across the computer science discipline to inculcate professional behaviour, strong ethical values, innovative research capabilities and leadership abilities which enable them to become successful entrepreneurs in this globalized world.

1. To nurture an **excellent learning environment** that helps students to enhance their problem solving skills and to prepare students to be lifelong learners by offering a solid theoretical foundation with applied computing experiences and educating them about their **professional, and ethical responsibilities**.
2. To establish **Industry-Institute Interaction**, making students ready for the industrial environment and be successful in their professional lives.
3. To promote **research activities** in the emerging areas of technology convergence.
4. To build engineers who can look into technical aspects of an engineering solution thereby setting a ground for producing successful **entrepreneur.**

INDEX OF THE CONTENTS

1. Introduction to the Lab

Lab 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

Course Outcome:

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.

2. LAB REQUIREMENTS

Hardware Detail

Intel i3/C2D Processor/2 GB RAM/500GB HDD/MB/Lan Card/ Key Board/ Mouse/CD Drive/15” Color Monitor/ UPS	24 Nos
LaserJet Printer	1 No

Software Detail

CentOS/Fedora Linux

3. LIST OF EXPERIMENTS

(As prescribed by G.G.S.I.P.U)

1. To determine the nature of roots of a quadratic equations, its input is triple of +ve integers (say a,b,c) 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.

2. To determine the type of triangle. Its input is triple of +ve integers (say a,b,c) 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

3. Perform robust case testing on Problem No. 1.

4. Perform robust case testing on Problem No. 2.

5. Create a test plan document for any application (e.g. Library Management System)

6. Experiment: Study of Any Testing Tool (Win Runner)

7. Experiment: Study of Any Test Management Tool (QA Complete)

8. Experiment: Automate the Test cases using Test Automation tool(using QA Complete)

9. Experiment: Learn how to raise and report Bugs using Bug tracking tool (Bugzilla,Jira using QA Complete)

10. Experiment: Study of any open source testing tool (Web Performance Analyzer/O STA

4.LIST OF EXPERIMENTS
(Beyond the syllabus)

1. Program to add two numbers, each number should be of one or two digits. Perform Adhoc testing.
2. To determine the nature of roots of a quadratic equations. Perform DD Path Testing Consider the following program segment that return roots of quadratic equation
   (1) int main ( )

   (2) {

   (3) int a, b, c, d, boolean = 0;

   (4) double D;

   (5) printf (“ Enter `a' coefficient :");

   (6) scanf ("%d", & a) ;

   (7) printf (" Enter `b' coefficient :″);

   (8) scanf ("%&d", & b);

   (9) printf (" Enter `c' coefficient :”);

   (10) scanf, ("%d″, & c) ;

   (11) if ((a > =0) && (a < = 00) && (b > = 0) && (b < =100) && (c > =0) && (c < =100)) {

   (12) boolean = 1;

   (13) if (a = = 0) {

   (14) boolean = -1;

   (15) }

   (16) }

   (17) if (boolean = = 1) {

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

   (19) if (d = = 0) {

   (20) printf ("roots are equal and are r1= r2 = %f - b/(2 * float)&));

   (21) }

```
(22) else if (d > 0) {

(23) D = sqrt (d);

(24) printf ("roots are real and are r1=%f and r2=%f; (-b - D)/(2 * a), (-b + D)/(2 * a));

(25) }

(26)  else {

(27) D = sqrt (-d) / (2 * a);

(28) printf ("roots are imaginary");

(29) }

(30) }

(31) else if (boolean = = -1) {

(32) printf ("Not a quadratic equation");

(33) }

(34) else {

(35) printf ("Invalid input range ...);

(36) }

(37) getch ( ):

(38) return 0;

(39) }
```

A. Draw the control flow graph for this program segment
B. Draw the DD Path Graph.
C. Calculation of Cyclomatic Complexity V(G) by three methods.
D. Determine the number of independent paths

3. Study of open sourceautomated testing suite for web applications (Selenium tool)

5. FORMAT OF THE LAB RECORD TO BE PREPARED BY THE STUDENTS

The front page of the lab record prepared by the students should have a cover page as displayed below.

NAME OF THE LAB

Paper Code

Font should be (Size 20", italics bold, Times New Roman)

Faculty name

Student name

Roll No.:

Semester:

Font should be (12", Times Roman)

Maharaja Agrasen Institute of Technology, PSP Area,

Sector – 22, Rohini, New Delhi – 110085

Font should be (18", Times Roman)

Index

Exp. no	Experiment Name	Date of performance	Date of checking	Marks	Signature

6. MARKING SCHEME FOR THE PRACTICAL EXAMS

There will be two practical exams in each semester.

i. Internal Practical Exam
ii. External Practical Exam

INTERNAL PRACTICAL EXAM

It is taken by the respective faculty of the batch.

MARKING SCHEME FOR THIS EXAM IS:

Total Marks: 40

Division of 10 marks per practical is as follows:

	Rubrics for : Laboratory (General)			
Sr No.	Experiment Component (LAC)	Max. Marks	Grading Rubrics	
			2 marks	1 mark
1	Practical Performance	2	Completeness of practical, exhibits proficiency in using different types of inputs.	Incomplete practical, unformatted, lacks comments, Demonstrates no proficiency.
2	Output and Validation	2	Output is free of errors and output is obtained. Demonstrates excellent understanding of the concepts relevant to the experiment.	Output contains few logical errors and/or no output is obtained. Demonstrates partial understanding of the concepts relevant to the experiment.
3	Attendance and Viva Questions Answered	4	1. Four marks for answering more than 75% questions. 2. Two marks for answering more than 50% questions. 3. One mark for answering less than 50% questions.	
4	Timely Submission of Lab Record	2	On time submission	Late submission

Each experiment will be evaluated out of 10 marks. At the end of the semester average of 8 best performed practical will be considered as marks out of 40.

EXTERNAL PRACTICAL EXAM

It is taken by the concerned lecturer of the batch and by an external examiner. In this exam student needs to perform the experiment allotted at the time of the examination, a sheet will be given to the student in which some details asked by the examiner needs to be written and at the last viva will be taken by the external examiner.

MARKING SCHEME FOR THIS EXAM IS:

Total Marks: 60

Division of 60 marks is as follows

1. Sheet filled by the student: 20
2. Viva Voice: 15
3. Experiment performance: 15
4. File submitted: 10

NOTE:

- Internal marks + External marks = Total marks given to the students
  (40 marks) (60 marks) (100 marks)

- Experiments given to perform can be from any section of the lab.

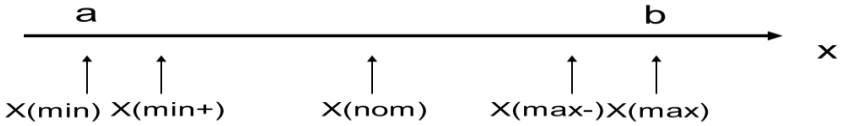
7. INSTRUCTIONS FOR EACH LAB EXPERIMENT

Experiment 1

Aim: To determine the nature of roots of quadratic equations from interval [1,100]. Perform BVA.

Performance Instructions:

BVA(Boundary value analysis)is a black box test design technique and it is used to find the errors at boundaries of input domain rather than finding those errors in the center of input The basic boundary value analysis technique can be generalized in two ways: by the number of variables, and by the kinds of ranges. Generalizing the number of variables is easy: if we have a function of n variables, we hold all but one at the nominal values, and let the remaining variable assume the min, min+, nom, max- and max values, and repeat this for each variable. Thus for a function of n variables, boundary value analysis yields 4n + 1 test cases.



In the above program consider the value 1(minimum), 2(just above Minimum), 50 (Nominal), 99(Just below Maximum) and 100(Maximum). Total No. of test cases will be 4*3+1=13

If a, b, and c denote the three integer in quadratic equation $a(x^2)+bx+c=0$ then Calculate discriminant d=(b*b)-4*a*c

if((a<1)||(b<1)||(c<1)||(a>100)||(b>100)||(c>100)) then "Invalid input"

if(a==0) then "Not a quadratic equation”

if (d==0) then "Roots are equal"

if(d<0) then "Imaginary roots"

otherwise "Real Roots"

Sample Inputs

A quadratic equation $a(x^2)+bx+c=0$ with input as three positive integers a, b, c having values ranging from an interval [1,100].

Sample Outputs

The program output may have one of the following :- Not a Quadratic equations, Real roots, Imaginary roots, Equal roots.

Result:

Test Case Id	a	b	c	Expected Output
1	50	50	1	Real roots
2	50	50	2	Real roots
3	50	50	50	Imaginary roots
4	50	50	99	Imaginary roots
5	50	50	100	Imaginary roots
6	1	50	50	Real roots
7	2	50	50	Real roots
-	-	-	-	-
-	-	-	-	-
1`3	-	-	-	-

Viva - Questions:

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

Q2. What is Boundary value analysis? Explain

Q3. What is Black Box testing technique?

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

Q5. Explain the weakness of boundary value analysis.

Q6. What are the limitation of boundary value analysis.

Experiment 2

Aim: To determine the type of triangle from interval [1,100]. Perform BVA.

Performance Instructions:

In the above program consider the value 1(minimum), 2(just above Minimum), 50 (Nominal), 99(Just below Maximum) and 100(Maximum). If a, b, and c denote the three integer sides, then the triangle property is mathematically stated as three inequalities: a < b + c, b < a + c, and c < a + b. If any one of these fails to be true, the integers a, b, and c do not constitute sides of a triangle. If all three sides are equal, they constitute an equilateral triangle; if exactly one pair of sides is equal, they form an isosceles triangle; and if no pair of sides is equal, they constitute a scalene triangle. Total No. of test cases will be 4*3+1=13 because number of input is the sides of triangle i.e. 3

Sample Inputs

Its input is triple of +ve integers (say a,b,c) and the values may be from interval[1,100] and

Sample Outputs:

The program output may be one of the following : Scalene, Isosceles, Equilateral, Not a Triangle.

Result:

Test Case Id	a	b	c	Expected Output
1	50	50	1	Isosceles
2	50	50	2	Isosceles
3	50	50	50	Equilateral
4	50	50	99	Isosceles
5	50	50	100	Not a triangle
-	-	-	-	-
-	-	-	--	-
-	-	-	-	-
13	-	-	-	-

Viva - Questions:

Q1. What is equivalence partitioning explain with example.

Q2. What is Test bed and Test data ?

Q3. Why does software have bugs?

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

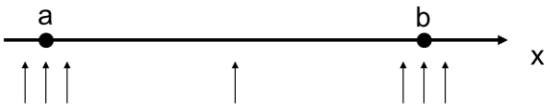
Q5. Describe the difference between validation and verification

Experiment 3

Aim: To determine the nature of roots of quadratic equations from interval [1,100]. Perform robust case testing

Performance Instructions:

Robustness testing is a simple extension of boundary value analysis: in addition to the five boundary value analysis values of a variable, in this technique we see what happens when the extreme are exceeded with a value slightly greater than the maximum (max+) and a value slightly less than the minimum (min-).



Test cases for a variable x, where a ≤ x ≤ b

Generalizing the number of variables is easy: if we have a function of n variables, we hold all but one at the nominal values, and let the remaining variable assume the min-, min, min+, nom, max- , max and max+ values, and repeat this for each variable. Thus for a function of n variables, Robustness testing yields 6n + 1 test cases.

In the above program consider the value 0(minimum-), 1(minimum), 2(just above Minimum), 50 (Nominal), 99(Just below Maximum), 100(Maximum) and 101(maximum+) Total No. of test cases will be 6*3+1=19

If a, b, and c denote the three integer in quadratic equation $a(x^2)+bx+c=0$ then

Calculate discriminant d=(b*b)-4*a*c

if((a<0)||(b<0)||(c<0)||(a>101)||(b>101)||(c>101)) then "Invalid input"

if(a==0) then "Not a quadratic equation”

if (d==0) then "Roots are equal"

if(d<0) then "Imaginary roots"

otherwise "Real Roots"

Sample Inputs

A quadratic equation $a(x^2)+bx+c=0$ with input as three positive integers a, b, c having values ranging from an interval [1, 100].

Sample Outputs

The program output may have one of the following: - Not Quadratic equations, Real roots, Imaginary roots, Equal roots.

Result:

Test Case Id	a	b	c	Expected Output
1	50	50	0	Real roots
2	50	50	1	Real roots
3	50	50	2	Real roots
4	50	50	50	Imaginary roots
5	50	50	99	Imaginary roots
6	50	50	100	Imaginary roots
7	50	50	101	Imaginary roots
8	0	50	50	Not a Quadratic equations
-	-	-	-	-
19	-	-	-	-

Viva - Questions:

Q1. What information does a test strategy capture?

Q2. How can you know when to stop testing?

Q3 What are the attributes of good test case?

Q4. What is robust case testing .How it is different from BVA?

Q5. Describe the Software Development Life Cycle

Experiment 4

Aim: To determine the type of triangle from interval [1,100]. Perform robust case testing

Performance Instructions:

In the above program consider the value 0(minimum-), 1(minimum), 2(just above Minimum), 50 (Nominal), 99(Just below Maximum) , 100(Maximum) and 101(maximum+). If a, b, and c denote the three integer sides, then the triangle property is mathematically stated as three inequalities: a < b + c, b < a + c, and c < a + b. If any one of these fails to be true, the integers a, b, and c do not constitute sides of a triangle. If all three sides are equal, they constitute an equilateral triangle; if exactly one pair of sides is equal, they form an isosceles triangle; and if no pair of sides is equal, they constitute a scalene triangle. Total No. of test cases will be 6*3+1=19 because number of input is the sides of triangle i.e. 3

Sample Inputs

Its input is triple of +ve integers (say a,b,c) and the values may be from interval[1,100] and

Sample Outputs:

The program output may be one of the following: Scalene, Isosceles, Equilateral, Not a Triangle.

Result:

Test Case Id	a	b	c	Expected Output
1	50	50	0	Not a triangle
2	50	50	1	Isosceles
3	50	50	2	Isosceles
4	50	50	50	Equilateral
5	50	50	99	Isosceles
6	50	50	100	Not a triangle
7	50	50	101	Not a triangle
-	-	-	-	-
19	-	-	-	-

Viva - Questions:

Q1. Give some advantage and disadvantage of functional test cases?

Q2. What are the characteristics of good test?

Q3. How do you measure test effectiveness and test efficiency

Q4. How much number of test cases is there in robust case testing?

Q5. What is the role of tester?

Experiment 5

Aim: Create a test plan document for any application (e.g. Library Management System)

Performance Instructions:

Test planning developed early phase of the software development and which includes followed by -:

Project Name -: Library Management System
Product Name -: Dues calculation and tracking books
Product Release Version -: 7.6.1

1] Introduction
2] Objectives and scope
3] Environment
4] Roles and Responsibility
5] Schedule
6] Testing strategies
7] Features to be tested
8] Features to be not tested
9] Entry and exit criteria
10] Dependency
11] Risk and mitigation's
12] Control procedures
13] Tools uses
14] Approval **Date -:** 26/3/2018
Prepared by -: Mr./Ms_____

Library Management Sample Test Plan

Prepared by: Ms / Mr. ……

Prepared Date: …/…/…

Introduction **-:**
The Library Management System application for assisting a librarian in managing library books. The system would provide basic set of features to add/update clients, add/update books, search for books, dues if any and manage check-in / checkout processes. This test plan is a basic guideline for future testing in the LMS.

Scope **-:**
The system would provide basic set of features to add/update members, add/update books,

dues if any and manage check in specifications for the systems based on the client's statement of need.

Environment Requirement -:
Hardware -: Three Dual Core or above machines needed
Software -: Microsoft Windows XP installed

Testing Strategies -:
1. Unit Testing :
Definition : Test smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation.
Participants/ Tested by : Developers
Methodology : Used for the Database test, records in each table, Basic function test, add a student, add a book, Network test

2. System and Integration Testing -:
Definition : Integration testing is the phase in software testing in which individual software modules are combined and tested as a group.
System integration testing (SIT) is a high-level software testing process in which testers verify that all related systems maintain data integrity and can operate in coordination with other systems in the same environment.
Participants/ Tested by : System Tester
Methodology : It is used for the Database test, Queries for insert, update, delete the records

3. Performance and Stress Testing -:
Definition : Determine how a system performs in terms of responsiveness and stability under a particular workload.
Stress testing tries to break the system under test by overwhelming its resources or by taking resources away from it
Participants/ Tested by : Tester
Methodology : It is used for the Database test, records in each table, Basic function test, Network test

4.User Acceptance Testing -:
Definition : Formal testing with respect to user needs, requirements, and business processes conducted to determine whether or not a system satisfies **Participants/ Tested by :** Users / End Users
Methodology : It is used for Whole System Test

5. Automated Regression Testing -:
Definition : Uncover new errors, or regressions, in existing functionality after changes have been made to a system, such as functional enhancements, patches or configuration changes.
Participants/ Tested by : Tester
Methodology : It is used for Whole System Test

Test Schedule -:

S. N	Particulars	Est. Start T.	Est. End T.	Actual Start T.	Actual End T.
1	Login				
2	Book Trans.				
3	Report Gen.				

Result: Library Management Sample Test Cases

Consider, library management system user is registered and logged in, then the detail for system test cases are as -:

Project Name-: Library Management Project Code-: Lib_Mgmt

Prepared by-: ---------- Prepared Date-:…/…/…

Reviewed by-: ---------- Review Date-:…/…/….

Total number of test cases =

Total number of test cases pass = —

Total number of test cases fail = —

Percentage of pass test cases = —

Percentage of pass test cases = —

For 1] Display available books on student name or Id

2] Dues calculation

3] Add category wise book

4] Search books by book name

5] Search book by author name

TC_ID	TC Desc.	TC Proc.	Input Data	Expected Res.	Actual Res.	Status
Lib_Mgmt1	Library Mgmt System Should be open	1] Enter student name or Student Id 2] Click on search	Student Name =" " or Student Id = " "	Available books should be displayed		
Lib_Mgmt2	Library Mgmt System Should be open	1] Enter student name or Student Id 2] Enter issued date 3] Enter received date 4] Click on calculate dues (Based policy to avail the books for specified days)	1] Student Name = " " or Student Id = " " 2] Issued date = " ' 3] Received date = " "	Dues should be calculated		
Lib_Mgmt3	Library Mgmt System Should be open	1] Enter Book name 2] Enter author name 3] Enter book category 4] Click on add book	Book name = " " Author name = " " Book category = " "	Category wise book should be added		
Lib_Mgmt4	Library Mgmt System Should be open	1] Enter Book name 2] Click on search	Book name = " "	Book should be displayed based on entered book		
Lib_Mgmt5	Library Mgmt System Should be open	1] Enter author name 2] Click on search	Author name = " "	Book should be displayed based on author name		

Viva - Questions:

Q1. Why should we test?

Q2. What is test Suite?

Q3. Define Test, Test Case.

Q4. How much testing is enough?

Q5.What is Test plan?

Experiment 6

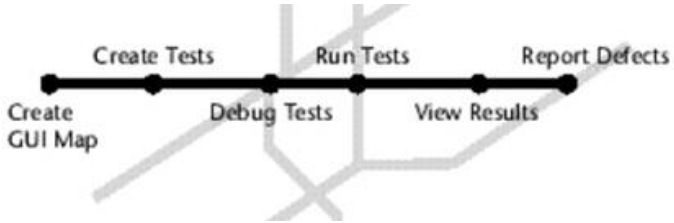
AIM: Study of Any Testing Tool (Win Runner)

Introduction to Win Runner

- WinRunner is Mercury's legacy automated testing tool.
- WinRunner is a test automation tool, designed to help customers save testing time and effort by automating the manual testing process.
- Automated testing with WinRunner addresses the problems by manual testing, speeding up the testing process.
- You can create test scripts that check all aspects of your application, and then run these tests on each new build.
- As WinRunner runs tests, it simulates a human user by moving the mouse cursor over the application, clicking Graphical User Interface (GUI) objects, and entering keyboard input.
- It create a summary report showing the test status

Winrunner Testing Process

Testing with WinRunner Involves six Stages



1) Create a GUI map: - WinRunner must learn to recognize the objects in an application in order to run tests - The preferred way to teach your objects depends on the GUI map mode
2) Create tests: - Winrunner writes scripts automatically when recording actions in application - One can program directly in Mercury interactive script language(TSL)
3) Debug tests: - You debug the test to check that they operate smoothly and without interruption
4) Run test: - Run test in verify mode t test your application - It compares the current data of application being tested to the expected data captured earlier - If any mismatch are found, winrunner captures them as actual results
5) View results: - After each run it displays result in report - The report details all the major events that occurred during the run such as checkpoint, error message, system message, user message
6) Report defects: - If a test run fails due to a defect it will report directly in report window

Features of WinRunner are: -

- Functional Regression Testing Tool
- Windows Platform Dependent
- Only for Graphical User Interface (GUI) based Application
- Based on Object Oriented Technology (OOT) concept
- Only for Static content
- Record/Playback Tool

Winrunner environment

• Windows - C++, Visual Basic, Java, PowerBuilder, Stingray, Smalltalk
• Web - Web Applications
• Other technologies - SAP, Siebel, Oracle, PeopleSoft, ActiveX

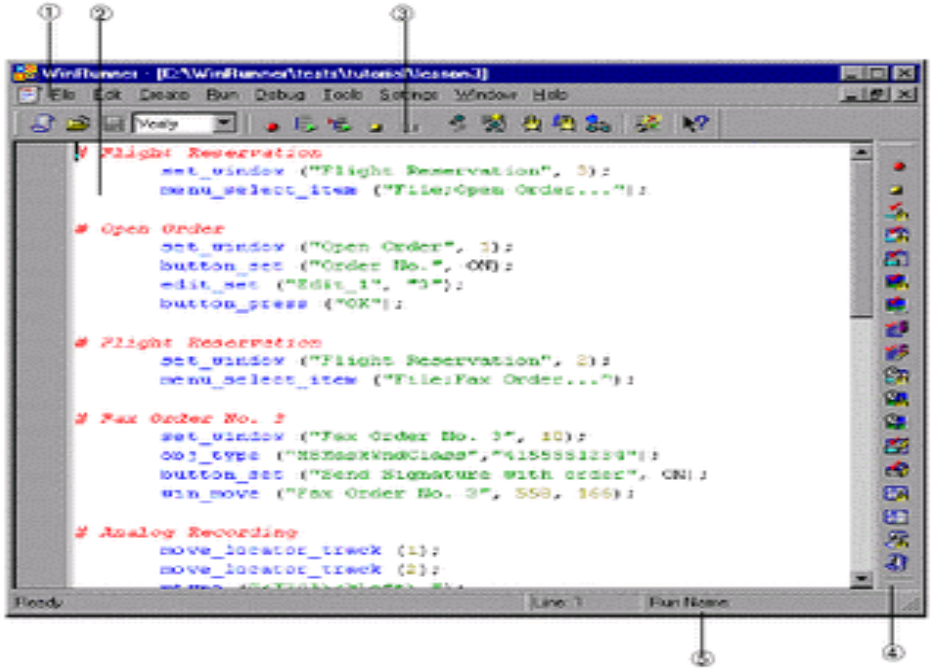
Exploring the WinRunner Window

To start WinRunner:

Choose Programs >WinRunner>WinRunner on the Start menu.

The first time you start WinRunner, the Welcome to WinRunner window opens. From the welcome window you can create a new test, open an existing test, or view an overview of WinRunner in your default browser.

Each test you create or run is displayed by WinRunner in a test window. You can open many tests at one time.



1 The WinRunner window displays all open tests.
2 Each test appears in its own test window. You use this window to record, program, and edit test scripts.
3 Buttons on the Standard toolbar help you quickly open, run, and save tests.
4 The User toolbar provides easy access to test creation tools.
5 The status bar displays information about selected commands and the current test run.

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

Viva - Questions:

Q1. How do you analyze test results in Winrunner tool and report the defects?

Q2.What are the different modes of recording in WinRunner?

Q3.What are the reasons that WinRunner fails to identify GUI object?

Q4. What do you mean by the logical name of the object?

Q5. What is the purpose of different record methods
1) Record

1) Pass up

2) As Object

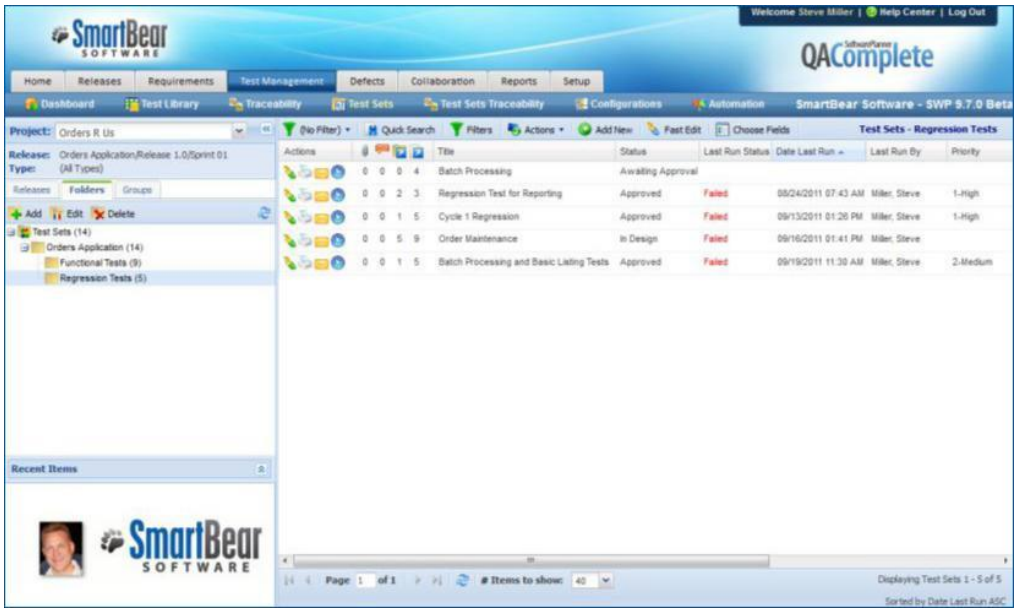
3) Ignore.

Experiment 7

AIM: Study of Any Test Management Tool (QA Complete)

TEST MANAGEMENT WITH QA COMPLETE

QA Complete gives the test team a single application for managing test cases, test environments, automated tests, defects and testing project tasks. QA Complete is the all-encompassing solution, providing visibility of test management process and ensuring the delivery of high quality software.



QA COMPLETE FEATURES

TEST CASE MANAGEMENT: Create test case suites to cover regression, smoke tests and standard release tests. Folder structure allows organization by release or feature. Simple test case structuring also allows for focused metrics and clear status reports.

TEST ENVIRONMENT MANAGEMENT: Using List Manager track all test environments and configuration. Link environments to individual test cases to see how effective test coverage is across different platforms, operating systems and devices.

DEFECT AND ISSUE MANAGEMENT: Track status and resolution progress of defects and issues for each release. Features, such as automatically creating defects on failed test cases, helps reduce time spent entering data and increase testers time spent running the tests.

TEST AUTOMATION INTEGRATION: Integrators with test automation tools like QTP and Test Complete allow you to track and report on the whole test management effort in one

place. With the ability to co-ordinate both manual and automated tests one can get the test data make release decision with ease.

BUG TRACKER INTEGRATION: Integrators with Jira, Bugzilla and other web based defect tracking tools allow tying together test effort in QA Complete with existing defect tracking tools implemented within organisation.

TEST PROJECT MANAGEMENT: Ensure that all test project tasks are tracked and monitored from writing the test plan to making the release decision. Variances can be tracked to help improve estimating and planning for future releases.

SHARED DOCUMENTS: Avoid the usual mess associated with storing test artifacts on different servers, in different directories and in different physical locations. Store all test documents in one central location to improve collaboration and communication within the test department.

REQUIREMENTS MANAGEMENT: Define requirements for each release and track the release each requirement is scheduled for. Workflow configuration allows tracking through user definable statuses. Integrated linking with test cases delivers clear requirements test coverage reports.

Viva - Questions:

Q1. What is bug leakage and bug release?

Q2. What is the difference between build and release?

Q3. What is the difference between the QA and software testing?

Q4. What are the automation challenges that SQA (Software Quality Assurance) team faces while testing?

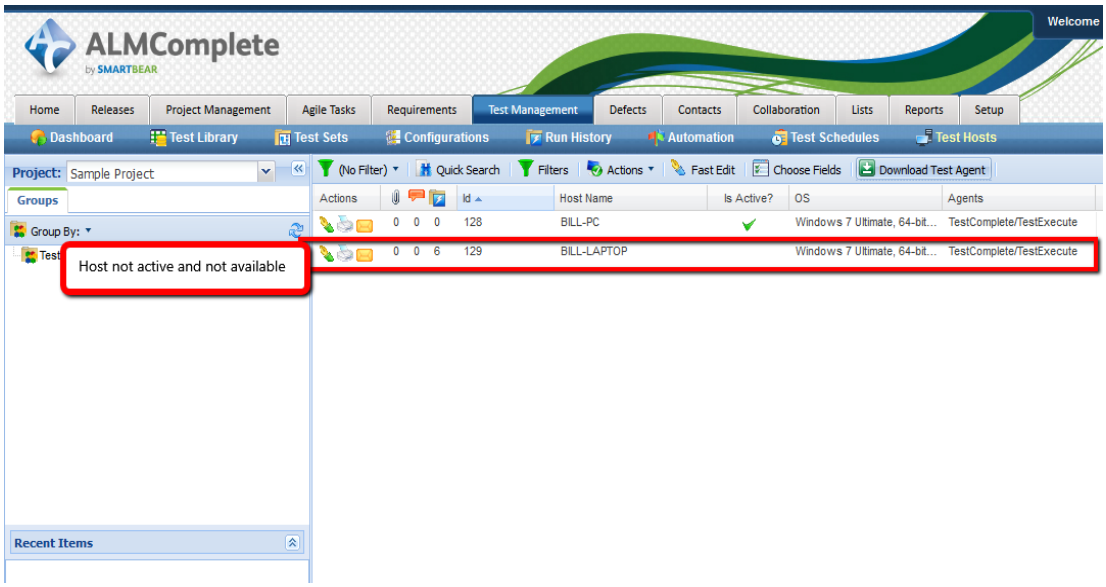
Q5. What is Agile testing and what is the importance of Agile testing?

Q6. Explain what should your QA documents should include?

Experiment 8

AIM: Automate the Test cases using Test Automation tool(using 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.

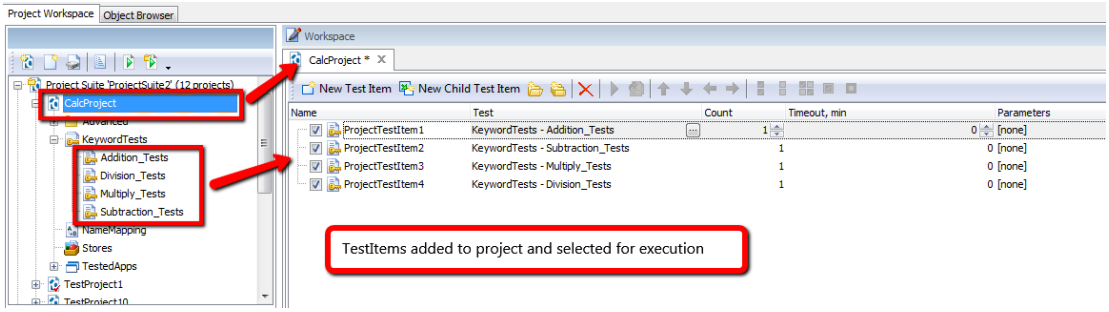
Step 3: Creating An Automate Test

This is a 4 stage process.

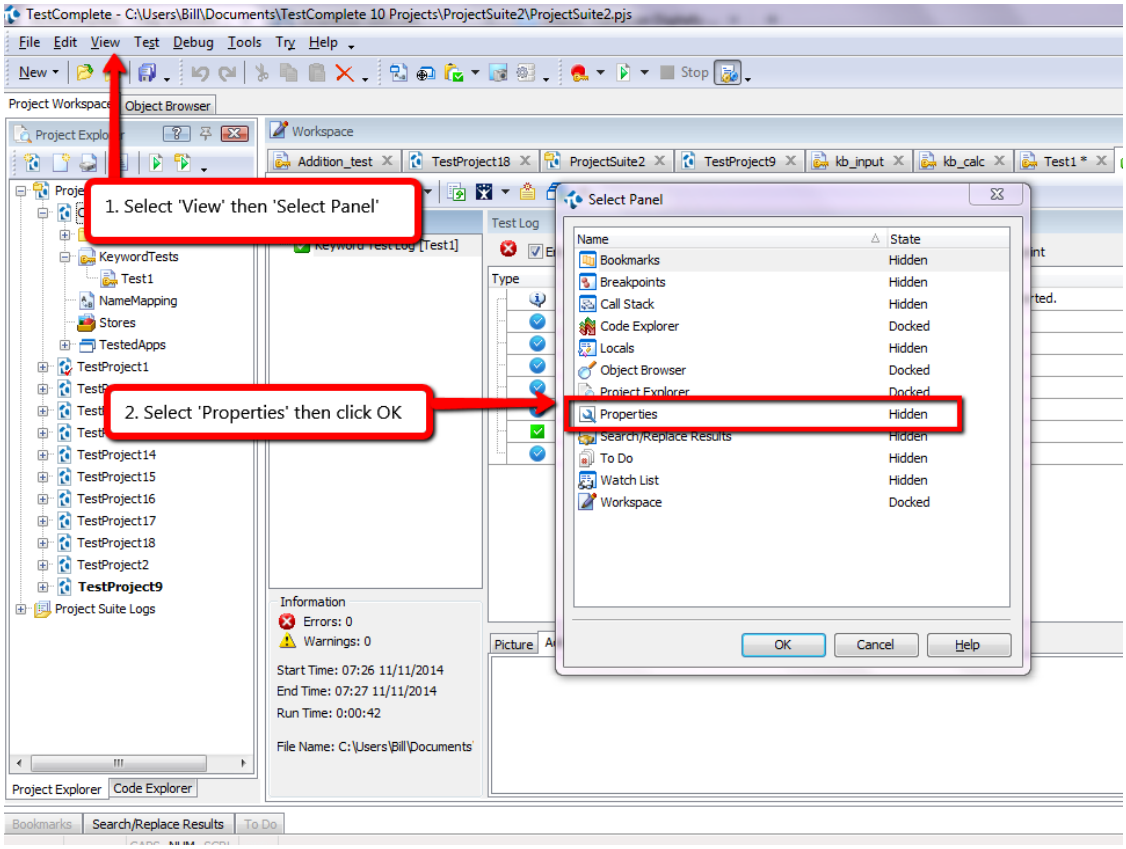
1. Package up the TestComplete project suite
2. Define the Automated Test in the QAComplete Test Library
3. Execution of Automated Tests – standalone
4. Exeuction of Automated Tests – as part of a Test Set

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

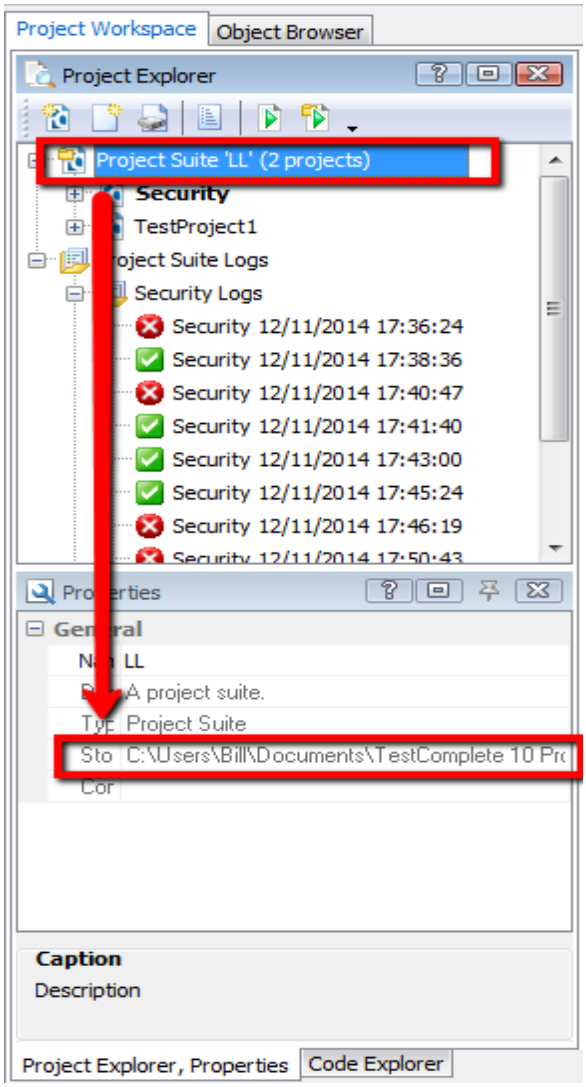
1) Make sure to define the 'Test Items' and enabled them within TestComplete project(s)



2) Find the 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.

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.

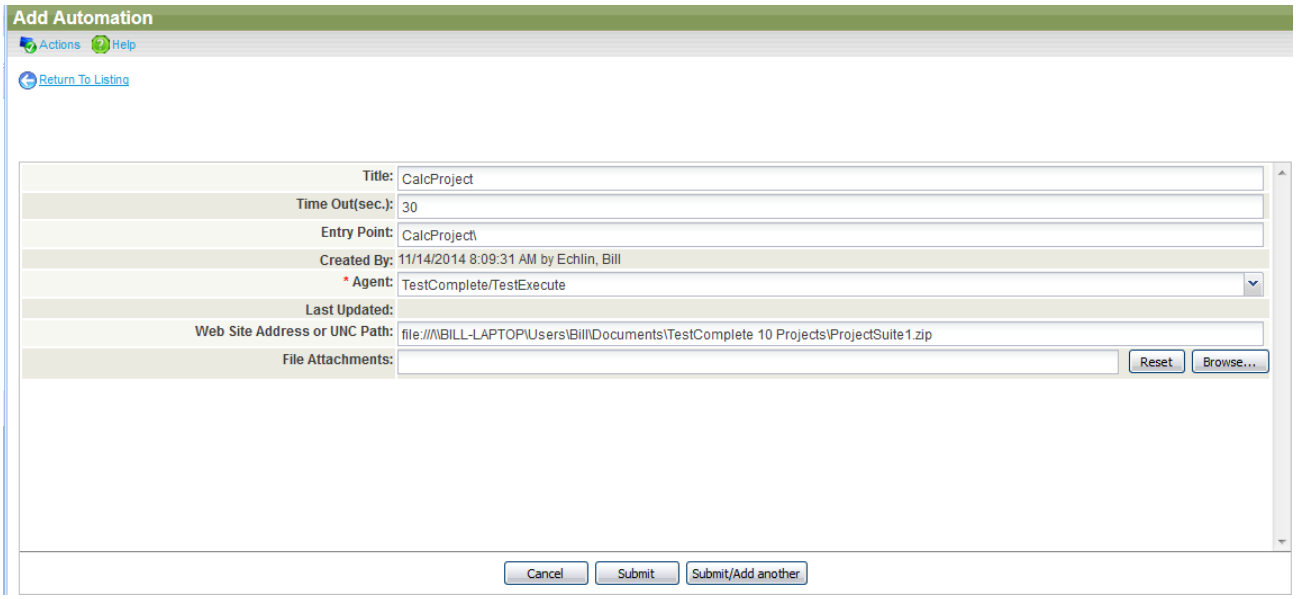
Agent: at the moment QAComplete only supports one type of test agent which is TestComplete/TestExecute. Other types of test agent are in the pipeline.

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)…



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

Viva - Questions:

Q1. What is Automation testing?

Q2. When will you automate a test?

Q3. What are the points that are covered while planning phase of automation?

Q4. What are the steps involved in the Automation Process?

Q5.When will you not automate testing?

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.

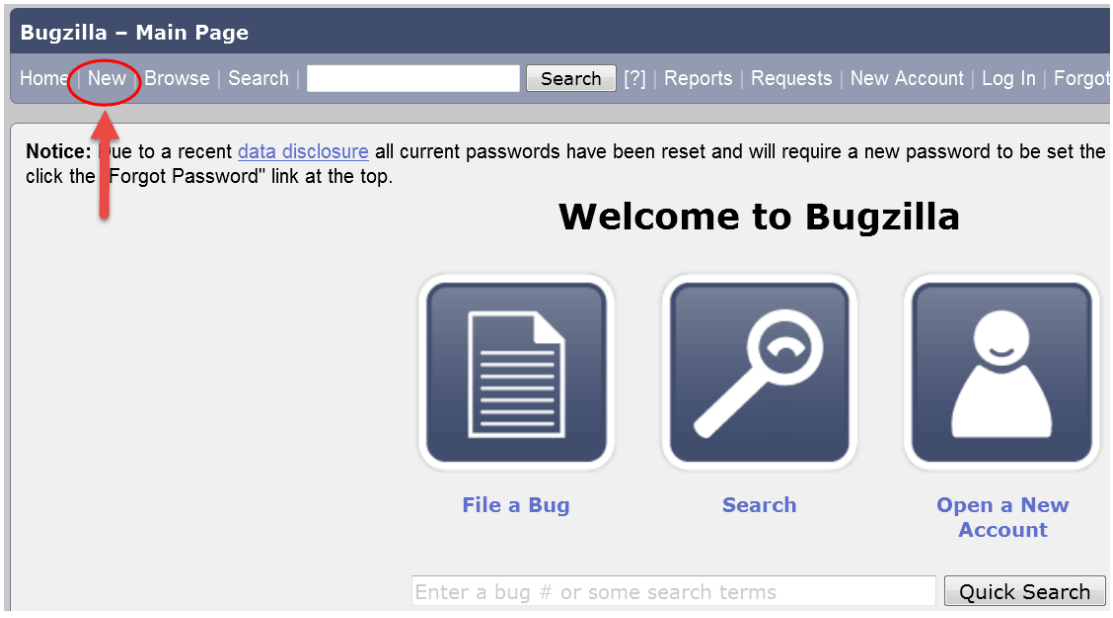
This open bug-tracker enables users to stay connected with their clients or employees, to communicate about problems effectively throughout the data-management chain.

Key features of bugzilla includes

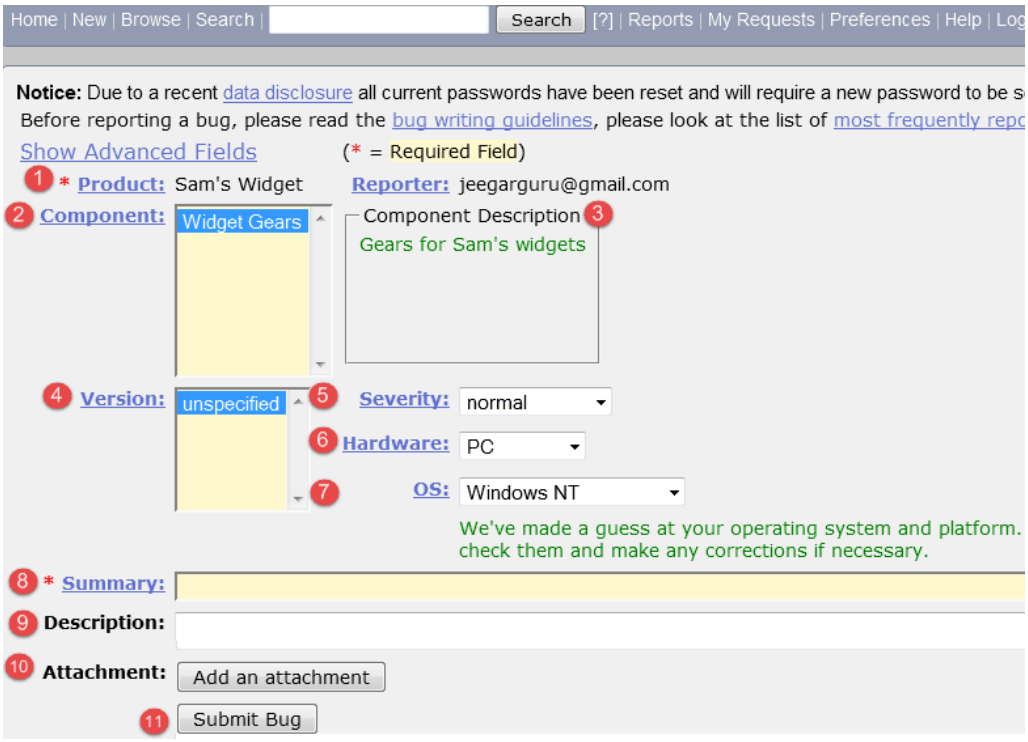
- Advanced search capabilities
- E-mail notifications
- Modify/file bugs by e-mail
- Time tracking
- Strong security
- Customization
- Localization

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

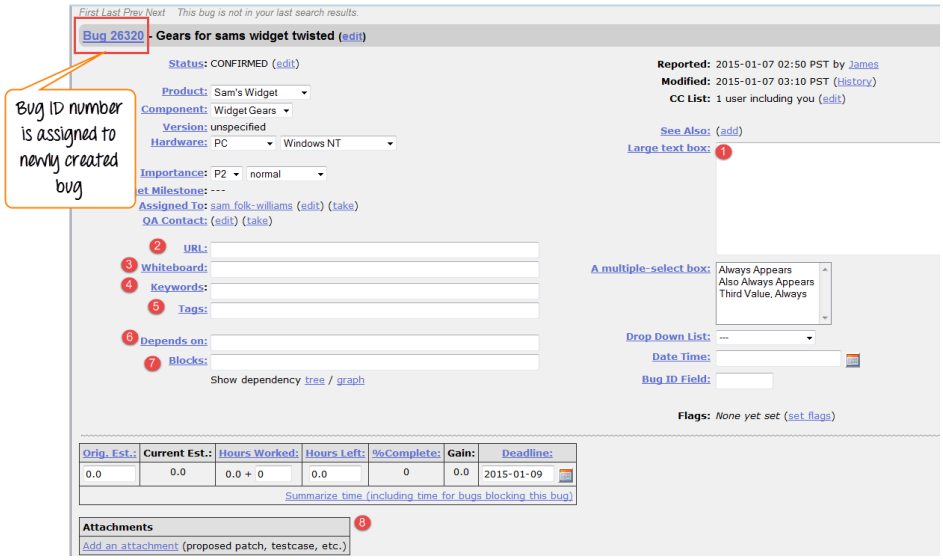


Step 2) In the next window

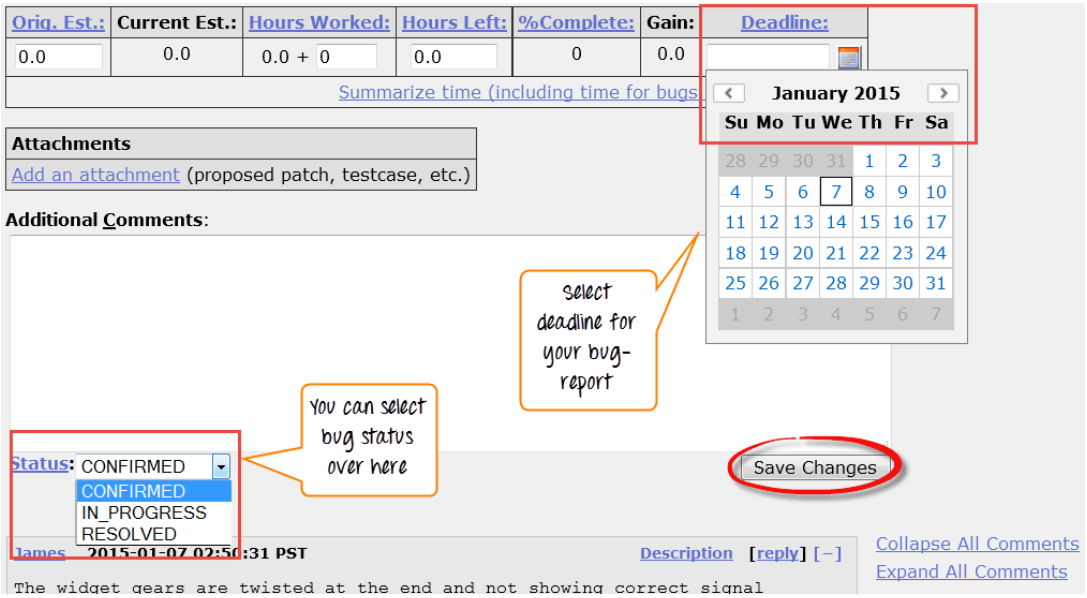
1. Enter Product
2. Enter Component
3. Give Component description
4. Select version,
5. Select severity
6. Select Hardware
7. Select OS
8. Enter Summary
9. Enter Description
10. Attach Attachment
11. Submit
12. 

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

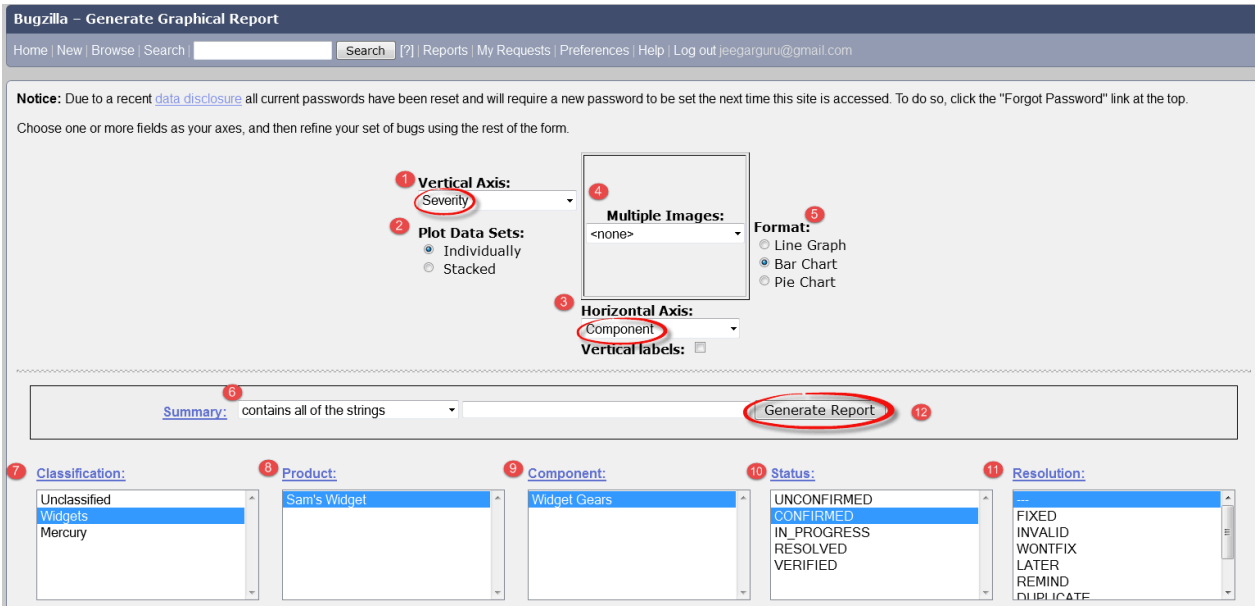


Step 5) in the same window if scroll down further then select deadline date and also status of the bug. Deadline in bugzilla usually gives the time-limit to resolve the bug in given time frame

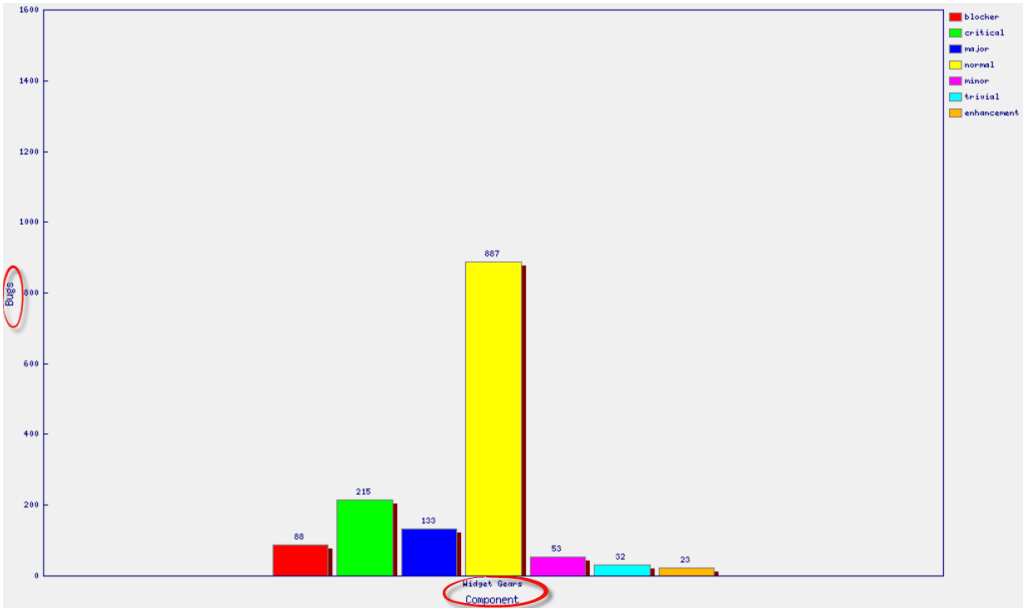


Create graphical reports

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.



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 - Questions:

Q1. Who benefits most from using the Bugzilla bug tracking tool?

Q2. Why is JIRA used?

Q3. What is the deployment environment for Bugzilla.

Q4. What is Bugzilla?

Q5. What are Bugzilla features?

Experiment 10

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

Introduction to web performance testing

The performance of web application can affects business. Top engineering organizations consider performance not as nice-to-have, but as a crucial feature of their products. Unfortunately, most engineering teams do not regularly test the performance and scalability of their infrastructure, and most lack the tools to properly do so. Although plenty of commercial tool options out there, for the right organization, free and open source tools may be a good alternative or the perfect complement to commercial tool set.

The goal of performance testing is to understand how your applications behave under heavy load conditions. To get started, you need to understand the baseline performance of your application and that the performance of each transaction is unique. For example, in an e-commerce application, a home page transaction is likely highly cached and very fast, whereas a checkout transaction is more complicated and must talk to a payment service, shipping service, etc. To ensure that users have a great experience, you must test the most common flows for your users and understand performance both in the browser and on the server.To get the job done, we'll need server-side, client-side, and performance tools.

Open STA (Open System Testing Architecture)

Open STA is the abbreviation for Open System Testing Architecture and it is built around COBRA. This GUI based testing tool is used by testers for load analyzing and load testing. This is a complex testing tools and is known to perform heavy test load for scripted HTTP and HTTPS. Here, the tests are performed using simple scripts, recordings and it also takes into account various results and statistics. During load tests, Open STA graphs resource utilization information from application servers, web servers, operating platforms and database servers that help in further analysis.

WAPT(Web Application Performance Tool) :

This web performance testing tool can be used for intranet applications and websites. WAPT is short for Web Application Performance Tool, and it acts as a scale or analyzing tool for measuring the output and performance of a web application and related interface. The tools help measure the performance of any web-related interface, web service or web application. You can use this tool to test the application performance in different environments and load conditions. The tool provides detailed information on the virtual users and its output to the users during load testing. It is considered by many to be one of the best and most cost-effective tools for analyzing the performance of web services. The WAPT tool can also test the compatibility of a web application with operating systems and browsers. It can be used

for testing the compatibility with Windows applications in certain cases. However, the tool only works on Windows operating systems.

Viva - Questions:

Q1. Why choose Open Source Performance Test tool?

Q2. Explain what is performance testing?

Q3. Mention different types of performance testing?

Q4. List out what are the common performance problem does user face?

Q5. List out some of the performance testing tool?

Q6. List out some common performance bottlenecks**?**

Experiments beyond the syllabus

Experiment 1

Aim: Program to add two numbers, each number should be of one or two digits. Perform Adhoc testing.

Performance Instructions:

Adhoc testing is an informal testing type with an aim to break the system. This testing is usually an unplanned activity. It does not follow any test design techniques to create test cases. In fact is does not create test cases altogether! This testing is primarily performed if the knowledge of testers in the system under test is very high. Testers randomly test the application without any test cases or any business requirement document. Ad hoc Testing does not follow any structured way of testing and it is randomly done on any part of application. Main aim of this testing is to find defects by random checking. Adhoc testing can be achieved with the testing technique called **Error Guessing.** Error guessing can be done by the people having enough experience on the system to "guess" the most likely source of errors.

Sample Inputs

Input is two numbers each should be one or two digits

```
#include<stdio.h>
void main()
{
int a, b, c;
scanf("%2d",&a);
scanf("%2d",&b);
  c=a+b;
printf("%2d\n",c);
}
```

Sample Outputs

The program output will be addition of two numbers

Result:

Test Case Id	Inputs (a, b)	Expected Output	Actual output
1	2,3	5	5
2	23, 45	68	68
3	-97, -99	-196	-2
10	-38, 99	61	5

Viva Questions

Q1. What exactly ad-hoc testing is?

Q2. At what situation it is prompt to implement ad-hoc testing?

Q3. What are the draw backs of ad-hoc testing?

Q4. What are the advantages of ad-hoc testing?

Experiment 2

Aim: To determine the nature of roots of a quadratic equations. Perform DD Path Testing

Consider the following program segment that returnroots of quadratic equation

```
(1) int main ( )

(2) {

(3) int a, b, c, d, boolean = 0;

(4) double D;

(5) printf ("nt Enter `a' coefficient :");

(6) scanf ("%d", & a) ;

(7) printf ("nt Enter `b' coefficient :");

(8) scanf ("%&d", & b);

(9) printf (" Enter `c' coefficient :");

(10) scanf, ("%d", & c) ;

(11) if ((a > =0) && (a < = 00) && (b > = 0) && (b < =100) && (c > =0) && (c < =100)) {

(12) boolean = 1;

(13) if (a = = 0) {

(14) boolean = -1;

(15) }

(16) }

(17) if (boolean = = 1) {

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

(19) if (d = = 0) {

(20) printf ("roots are equal and are r1= r2 = %f - b/(2 * float)&));

(21) }

(22) else if (d > 0) {
```

```
(23) D = sqrt (d);

(24) printf ("roots are real and are r1=%f and r2=%f; (-b - D)/(2 * a), (-b + D)/(2 * a));

(25) }

(26)  else {

(27) D = sqrt (-d) / (2 * a);

(28) printf ("roots are imaginary");

(29) }

(30) }

(31) else if (boolean = = -1) {

(32) printf ("Not a quadratic equation");

(33) }

(34) else {

(35) printf ("Invalid input range ...);

(36) }

(37) getch ( ):

(38) return 0;

(39) }
```

A. Draw the control flow graph for this program segment

B. Draw the DD Path Graph.

C. Calculation of Cyclomatic Complexity V(G) by three methods.

D. Determine the number of independent paths

Performance Instructions:

White box testing: White box testing sometimes called open box testing or glass box testing or clear box testing or structural testing. In white box testing test cases are derived from the source code internals of the software, specifically including branches, individual conditions, and statements. It is concerned with the level up to which test cases exercise or cover the

logic (source code) of the program. By examining and testing the program code white box testing tests the external functionality of the code. In White Box testing the tester must have the complete knowledge about the internal structure of the source code. Using white box testing methods the tester can derive following test cases:

1. Test case that ensures that all independent path have been exercised at least once.
2. Test cases that checks all the decision on their true or false sides.
3. Test cases that execute all loops at their boundaries and within their operational bound.
4. Test cases that exercise internal data structures to ensure their validity.

Following are the white box testing techniques:

- Basis Path testing
- Data Flow testing
- Mutation Testing

Basis Path testing

With the help of basis path testing, test case designer can derive a logical complexity measure of a procedural design and can use this measure to find the basis set of execution paths. Test cases that are derived to exercise the basis set ensures that every statement in the program will execute at least one time during testing.

Steps for basis path testing

- construct the flow graph from the source code or flow chart
- Identify independent path in the flow graph
- Calculate cyclomatic complexity, V(G).
- Design the test cases.

Flow graph

Flow graph of a program can be represented using a graphical representation known as a "Flow Graph".A Flow graph is a directed graph in which nodes are either entire statements or fragment of statements. Edges represent the flow of control.Flow graph depicts the logical control flow using the notation given in fig 2.4

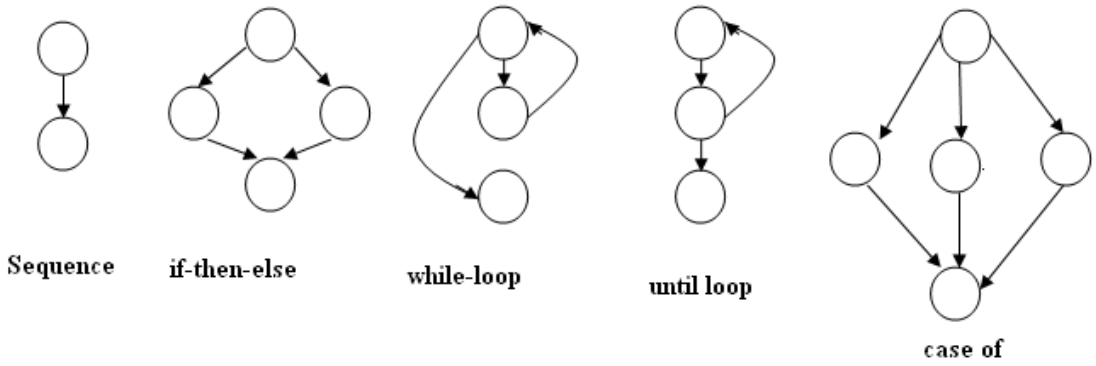


Fig:Structured programming constructs

Cyclomatic complexity can be computed in one of the tree ways:

1. The number of regions in a flow graph corresponds to the cyclomatic complexity.
2. Cyclomatic complexity V(G) of a flow graph G, is defined as
   V(G)= E-N+2

   Where E is the Number of edges in the flow graph

   N is th number of nodes in flow graph.

3. Cyclomatic complexity V(G) of a flow graph G, can also be defined as
   V(G) = P+1

Where P is the number of predicate node in the flow graph

Independent path : Number of independent path = V(G) which makes a basis set.

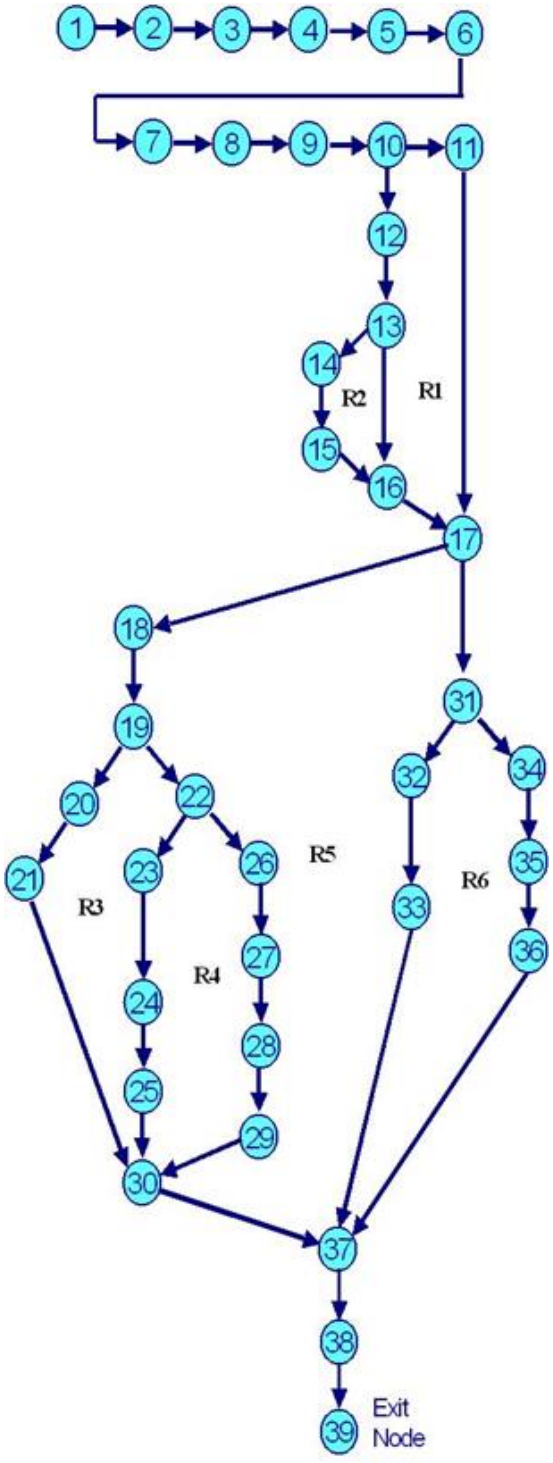
We must execute these paths at least once in order to test the program thoroughly.

Sample Input: input will be three integers 50, 50, 2

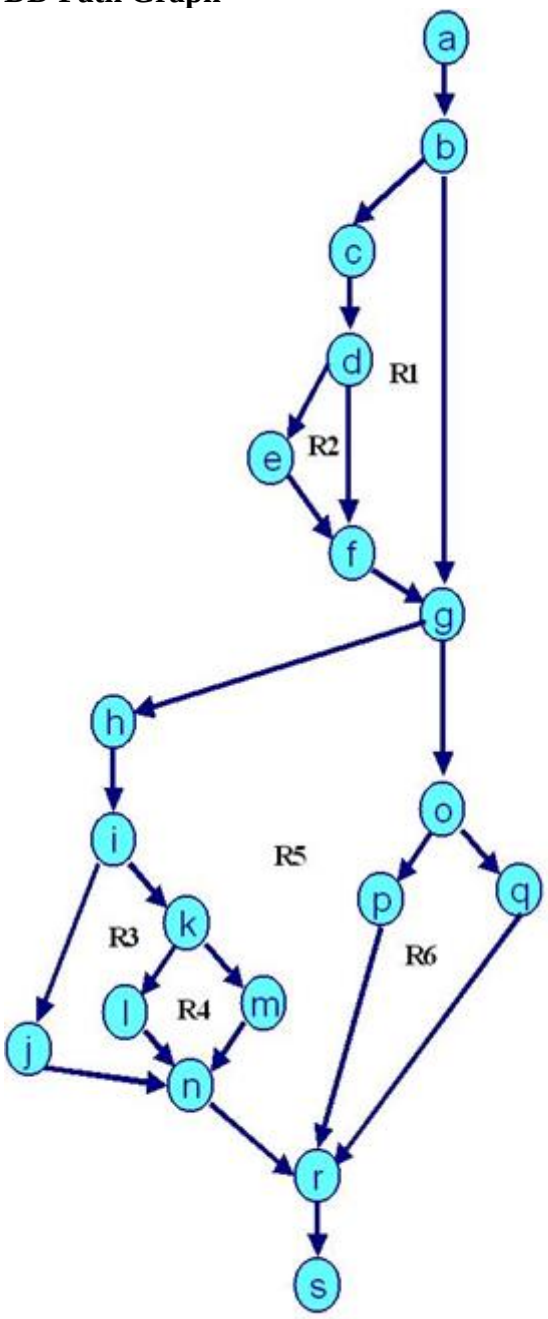
Sample Output: roots are real root

Result:

A. Control flow graph for the program segment is as follows:



B. DD Path Graph



Decision table for the above program :

Nodes in Flow Graph	**Corresponding Nodes of DD Path Graph**	**Justification**
1- 9	a	Are Sequential Nodes
10	b	Decision Nodes

11	c	Decision Nodes
12, 13	d	Sequential Nodes
14	e	Two Edges Combined
15, 16, 17	f	Sequential Nodes
18	g	Decision Nodes
19	h	Decision Nodes
20, 21	i	Sequential Node
22	j	Decision Node
23, 24	k	Sequential Node
25, 26, 27	I	Sequential Nodes
28	m	Three Edges Combined
29	n	Decision Node
30, 31	o	Sequential Node
32, 33, 34	p	Sequential Nodes
35	q	Three edges Combined
36, 37	r	Sequential Exit Nodes

C. Cyclomatic Complexity V(G) by three methods

Method 1: V(G) = e - n + 2 (Where e are edges & n are nodes)

V(G) = 24- 19+ 2 = 5 + 2 = 7

Method 2: V(G) = P + 1 (Where P is No. of predicate nodes with out degree = 2)

V(G) = 6 + 1 = 7 (Nodes d, b, g, I, o & k are predicate nodes with 2 outgoing edges)

Method 3: V(G) = Number of enclosed regions + 1 = 6+1=7
(Here R1, R2, R3, R4, R5 & R6 are the enclosed regions and 1 corresponds to one outer region)

Cyclomatic Complexity =V(G) = 7 and is same by all the three methods**.**

D. Draw independent paths
Number of independent paths = V(G)=7

Path 1:

a b f g n p q r

Path 2:

a b f g n o q r

Path 3:

a b c e g n p q r

Path 4:

a b c d e g n o q r

Path 5:

a b f g h i m q r

Path 6:

a b f g h i k m q r

Path 7:

a b f g h j l m q r

Viva Questions:

Q1. Difference between white box and black box testing techniques

Q2. Define cyclomatic complexity

Q3. What is DD Path Testing.

Q4. Why we do White Box Testing?

Q5. Draw the flow graph for the following cade

```
Void fun(float a, float b *, int c)
{
float d = a;
if(d > 0.01)
d = a;
else
d = c;
for(itn I = 0; I < d; i++)
b[i]= b[i]*c;
cout<<b[i];
}
```

Experiment 3

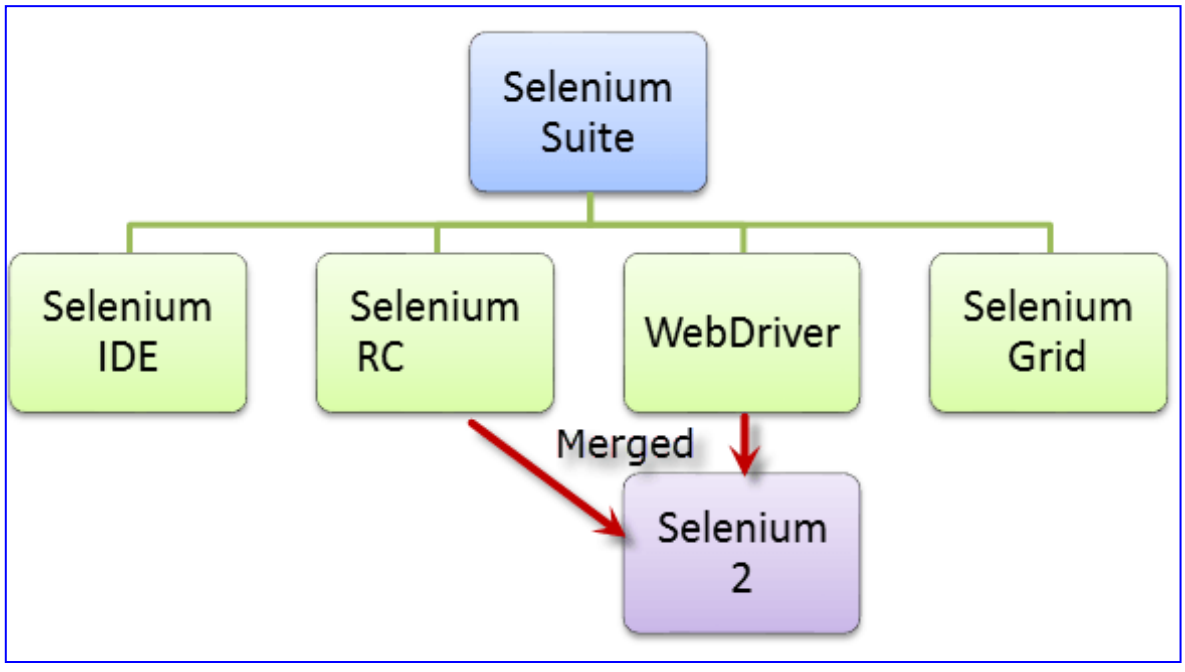
AIM: Study of open sourceautomated testing suite for web applications (Selenium tool)

Introduction to Selenium tool

Selenium is a free (open source) automated testing suite for web applications across different browsers and platforms. It is quite similar to HP Quick Test Pro (QTP) only that Selenium focuses on automating web-based applications.

Selenium is not just a single tool but a suite of software's, each catering to different testing needs of an organization. It has four components.

- Selenium Integrated Development Environment (IDE)
- Selenium Remote Control (RC)
- WebDriver
- Selenium Grid



Introduction to Selenium IDE

Selenium IDE (Integrated Development Environment) is the simplest tool in the Selenium Suite. It is a Firefox add-on that creates tests very quickly through its record-and-playback functionality. This feature is similar to that of QTP. It is effortless to install and easy to learn.

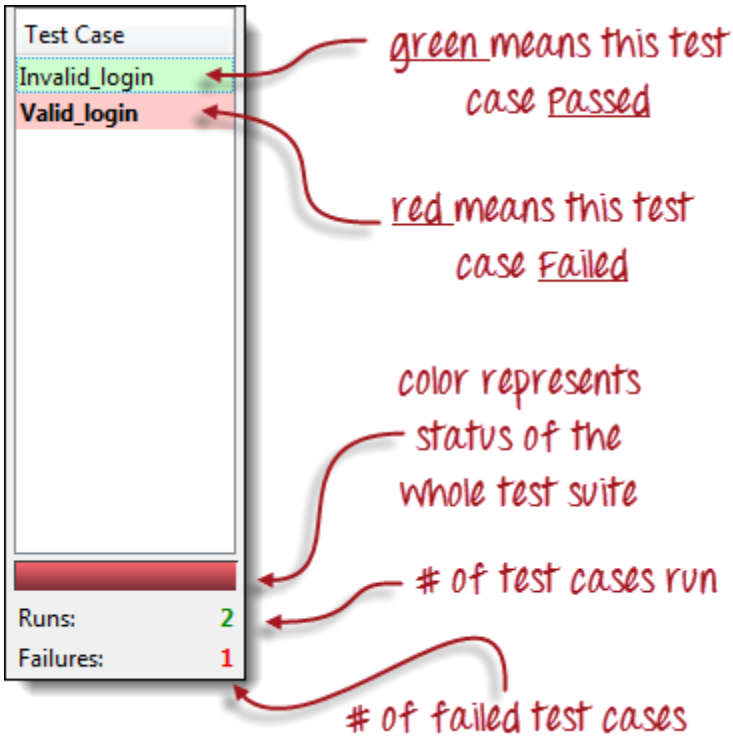
Because of its simplicity, Selenium IDE should only be used as a prototyping tool - not an overall solution for developing and maintaining complex test suites.

Selenium IDE supports autocomplete mode when creating tests. This feature serves two purposes:

- It helps the tester to enter commands more quickly.
- It restricts the user from entering invalid commands.

Features of Selenium IDE

Test Case Pane



- In Selenium IDE, you can open **more than one test case at a time**.
- **The test case pane shows you the list of currently opened test cases.**
- When you open a test suite, the test case pane will **automatically list all the test cases** contained in it.
- The test case written in **bold font** is the **currently selected test case**
- After playback, **each test case is color-coded** to represent if it passed or failed.
    - Green color means "Passed."
    - Red color means "Failed."
- At the bottom portion is a summary of the number of test cases that were run and failed.

Viva Questions:

Q1. What is Selenium IDE?

Q2. What is the Test step Syntax?

Q3. What are the Advantages of Selenium IDE?

Q4. How to install Selenium IDE?

Q5. How to create Test cases using Selenium IDE?

Q6. How to execute Selenium Test case step by step?

8. Sample Viva – Questions

Q1. Explain Cyclomatic Complexity.
Q2. Explain Boundary value testing and Equivalence testing with some examples.
Q3. What is Security testing?
Q5. What is AUT?
Q6. What is Defect Leakage?
Q7. What are the contents in an effective Bug report?
Q8. What is Bug Life Cycle?
Q9. What is Error guessing and Error seeding?
Q10. What is the difference between Bug, Error and Defect?
Q11. What is Test bed and Test data?
Q12. What is Negative testing?
Q13. Explain Load, Performance and Stress Testing with an Example.
Q14. What are SDLC and STLC ? Explain its different phases.
Q15. What is Ad-hoc testing?
Q16. Describe bottom-up and top-down approaches in Regression Testing.
Q17. What is the difference between structural and functional testing?
Q18. What is Re- test? What is Regression Testing?
Q19. What is UAT testing? When it is to be done?
Q20. What are the basic solutions for the software development problems?
Q21. What are the common problems in the software development process?
Q22. Why does software have bugs?
Q23. What software testing types can be considered?
Q24. How do you decide when you have 'tested enough'?
Q25. Describe the Software Development Life Cycle
Q26. Describe the difference between validation and verification
Q27. What is the difference between QA and testing?
Q28. What is quality assurance?
Q29. What is the purpose of the testing?
Q30. What's an 'inspection'?
Q31. What are 5 common problems in the software development process?
Q32. How can it be known when to stop testing?