# INFORMATION SECURITY LAB

***Paper code : ETCS-451***

# Faculty Name: Student Name: Anupam Gupta

Ms. Prerna Roll Number: 00196407220

Semester: 7th



Maharaja Agrasen Institute of Technology, PSP Area,

Sector – 22, Rohini, New Delhi – 110085

# INDEX

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Exp No. | Experiment Name | Date of performance | Date of checking | Marks (10) | Signature |
| 1 | 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 | * 1. 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. |  |  |  |  |
| 4 | * 1. Perform robust case testing on the above problem. |  |  |  |  |
| 5 | Create a test plan document for any application (e.g. Library Management System). |  |  |  |  |
| 6 | Study of Any Testing Tool (Win Runner). |  |  |  |  |
| 7 | Study of Any Test Management Tool (QA Complete) |  |  |  |  |
| 8 | Automate the Test cases using Test Automation tool(using QA Complete) |  |  |  |  |

# INDEX

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Exp No. | Experiment Name | Date of performance | Date of checking | Marks (10) | Signature |
| 9 | 1. Learn how to raise and report Bugs using Bug tracking tool (Bugzilla, Jira using QA Complete). |  |  |  |  |
| 10 | * 1. Study of any open source testing tool (OSTA). |  |  |  |  |
| 11 | Program to add two numbers, each number should be of one or two digits. Perform Adhoc testing. |  |  |  |  |
| 12 | * 1. To determine the nature of roots of a quadratic equations. Perform DD Path Testing |  |  |  |  |
| 13 | Study of open source automated testing suite for web applications (Selenium tool). |  |  |  |  |
| 14 |  |  |  |  |  |
| 15 |  |  |  |  |  |
| 16 | A |  |  |  |  |

**Experiment – 1 & 3**

**Aim:**

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. Perform robust case testing.

**Theory:**

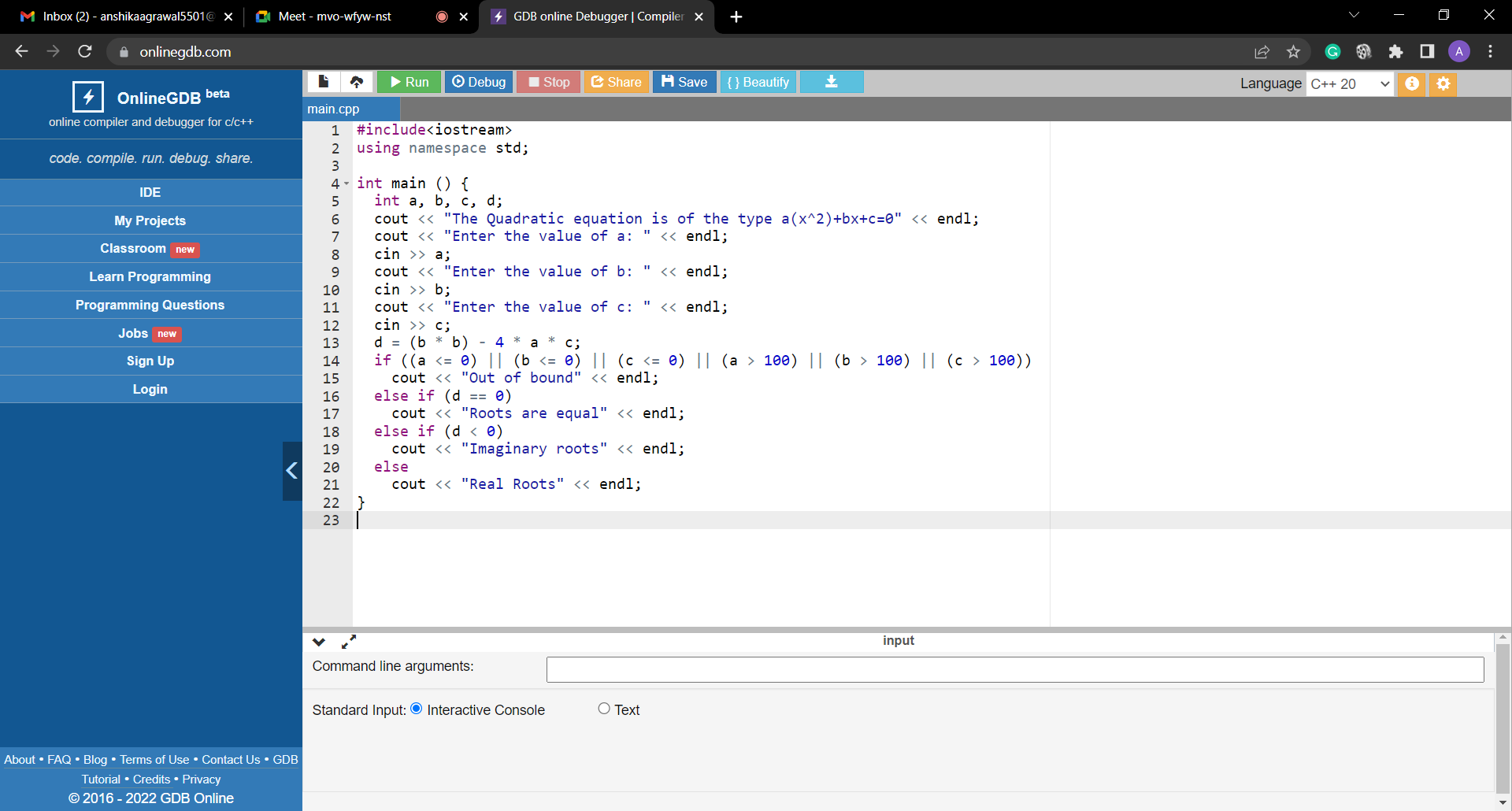
**Boundary Value Analysis**

1. Boundary Value Analysis is black box testing technique which is used to identify errors at boundaries rather than finding those that exist in the centre of the input domain.
2. Boundary value analysis is a technique to test boundary value between valid and invalid partitions in test case design.
3. Boundary Value Analysis is often called as a part of the Stress and Negative Testing.
4. For each variable we check-
   1. i) Minimum value.
   2. ii) Just above the minimum.
   3. iii) Nominal Value.
   4. iv) Just below Max value
   5. v) Max Value

**Limitation of Boundary Value Analysis:**

* It works well when the product is under test.
* It cannot consider the nature of the functional dependencies of variables.
* BVA is quite rudimentary.

**Code:**



**TEST CASES:**

A boundary value analysis has a total of 4\*n+1 distinct test cases, where n is the number of variables in a problem. Here we have to consider all the three variables and design all the distinct possible test cases. We will have a total of 13 test cases as n = 3.

* Roots are real if (b2 – 4ac) > 0
* Roots are imaginary if (b2 – 4ac) < 0
* Roots are equal if (b2 – 4ac) = 0
* Equation is not quadratic if a = 0

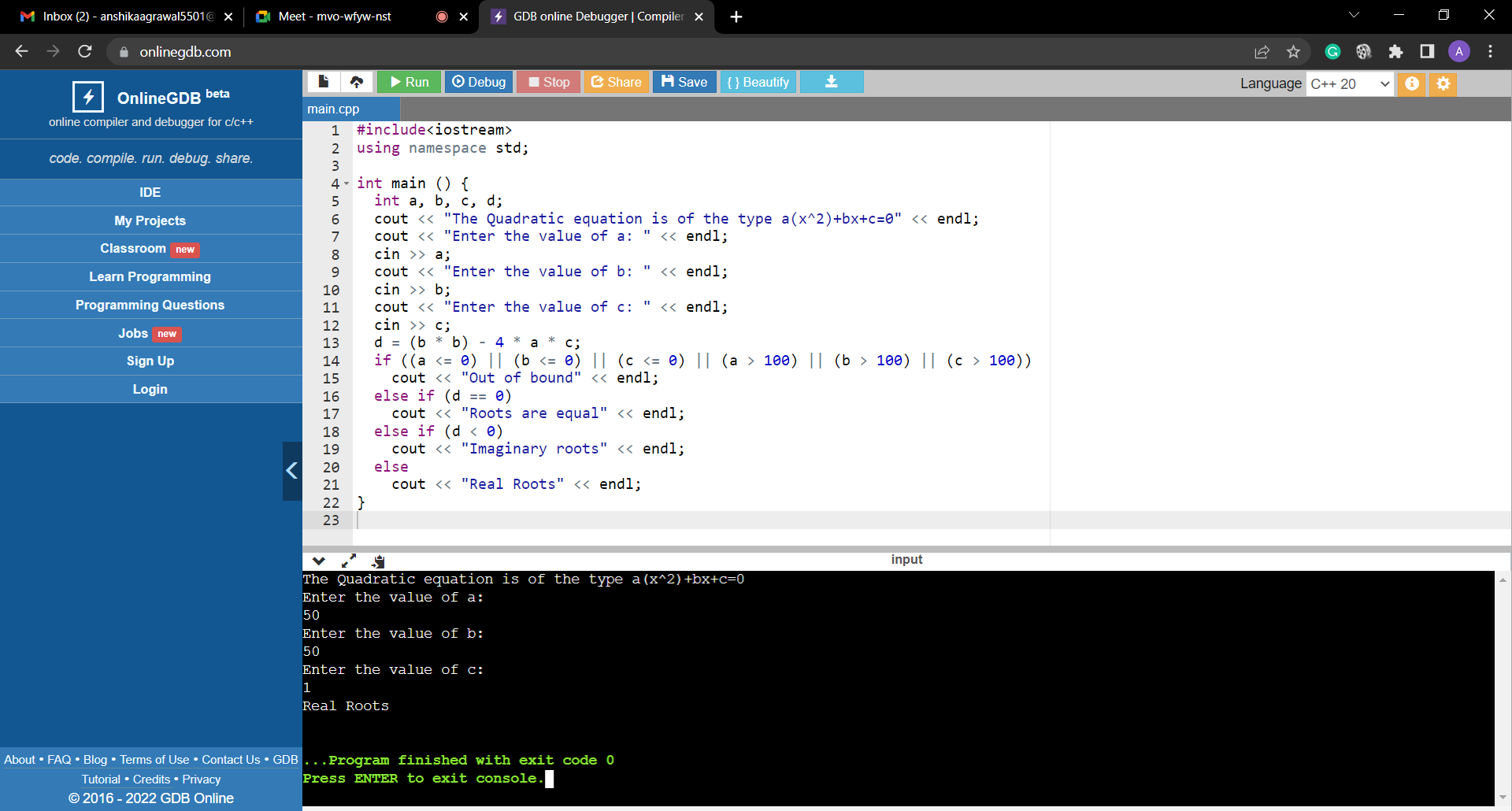
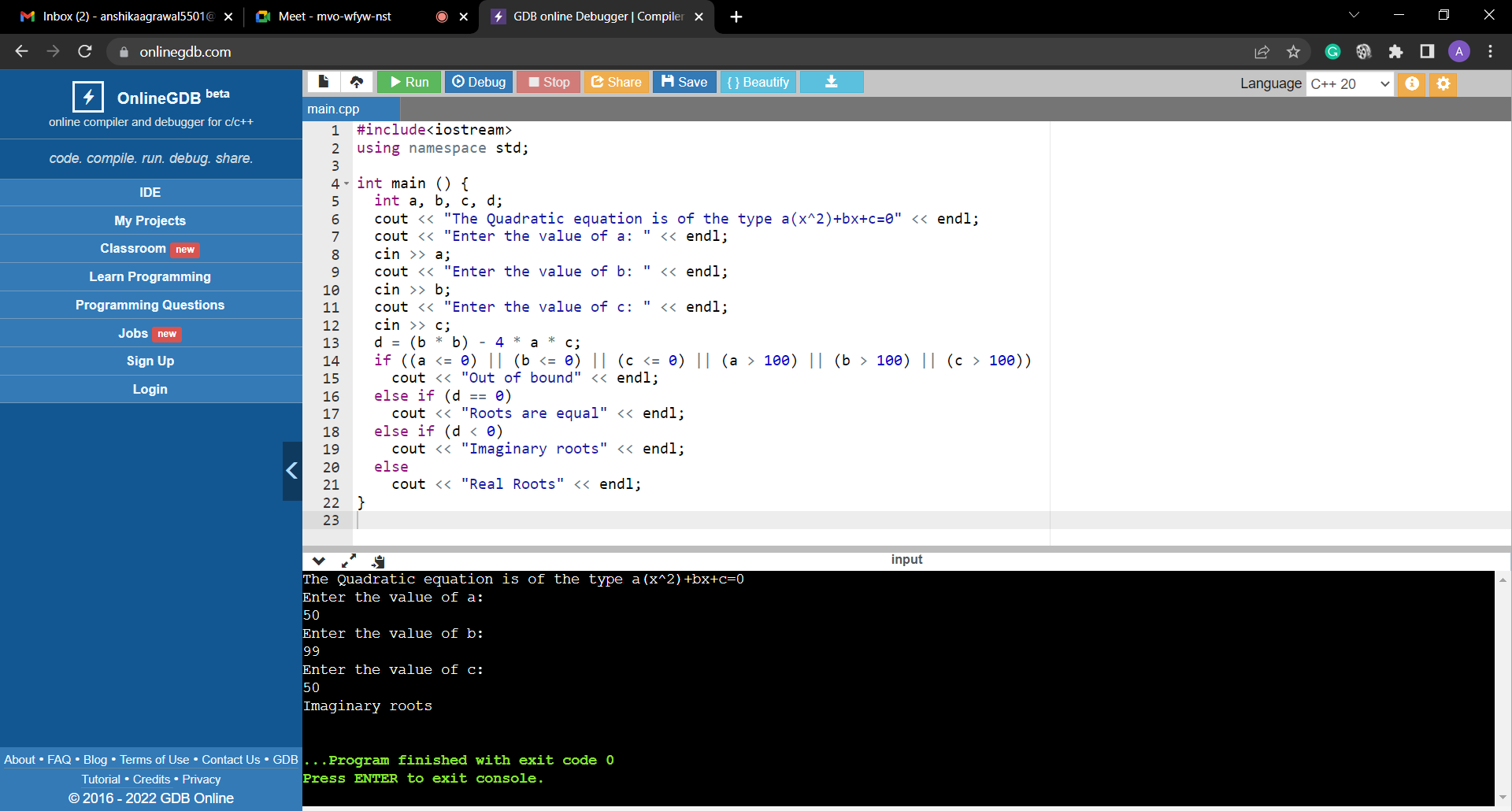
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Testcase** | **a** | **b** | **c** | **Expected**  **Output** | **Observed**  **Output** |
| 1 | 1 | 50 | 50 | Real Roots | Real Roots |
| 2 | 2 | 50 | 50 | Real Roots | Real Roots |
| 3 | 50 | 50 | 50 | Imaginary Roots | Imaginary Roots |
| 4 | 99 | 50 | 50 | Imaginary Roots | Imaginary Roots |
| 5 | 100 | 50 | 50 | Imaginary Roots | Imaginary Roots |
| 6 | 50 | 1 | 50 | Imaginary Roots | Imaginary Roots |
| 7 | 50 | 2 | 50 | Imaginary Roots | Imaginary Roots |
| 8 | 50 | 99 | 50 | Imaginary Roots | Imaginary Roots |
| 9 | 50 | 100 | 50 | Equal Roots | Equal Roots |
| 10 | 50 | 50 | 1 | Real Roots | Real Roots |
| 11 | 50 | 50 | 2 | Real Roots | Real Roots |
| 12 | 50 | 50 | 99 | Imaginary Roots | Imaginary Roots |
| 13 | 50 | 50 | 100 | Imaginary Roots | Imaginary Roots |

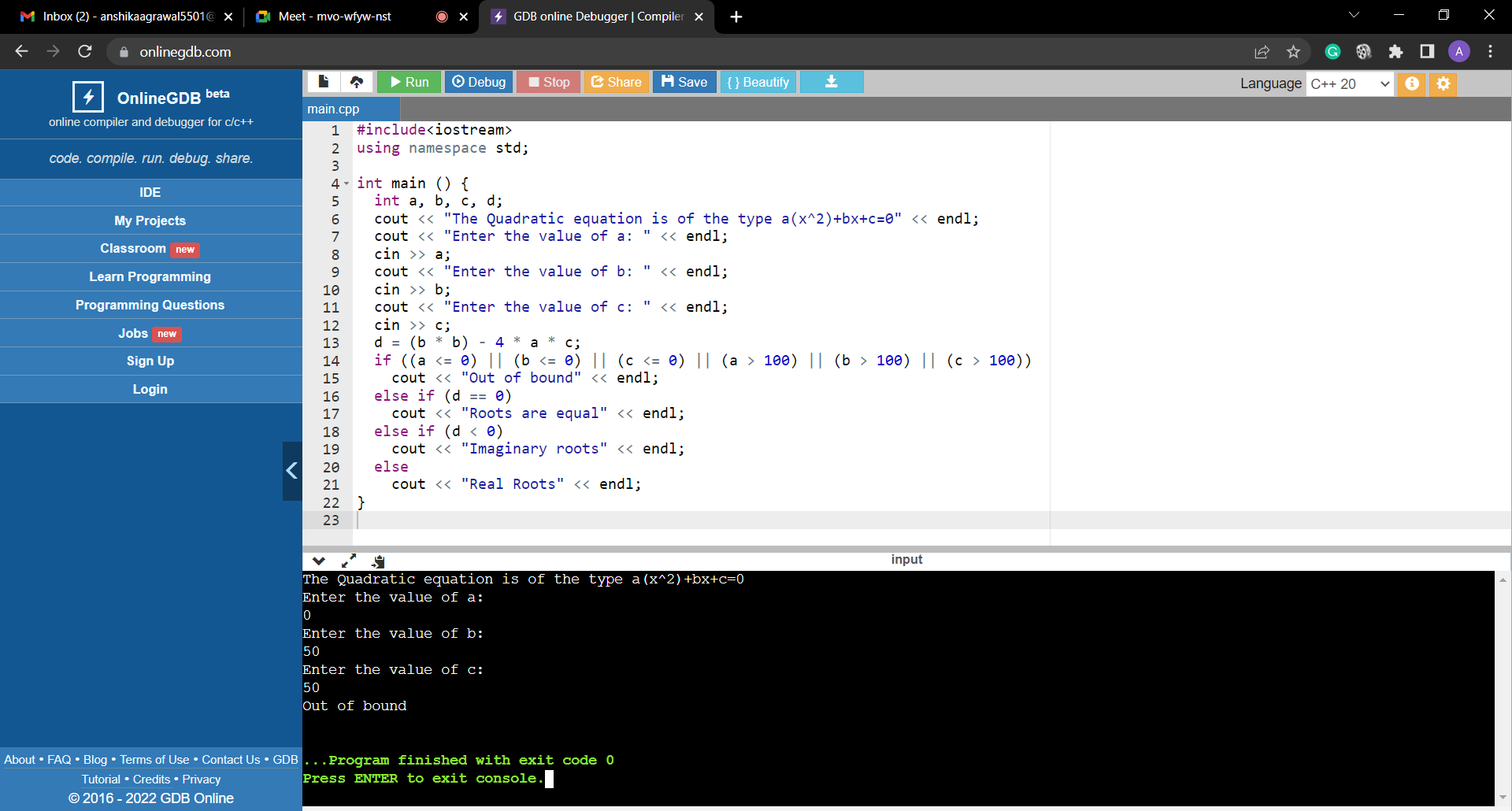
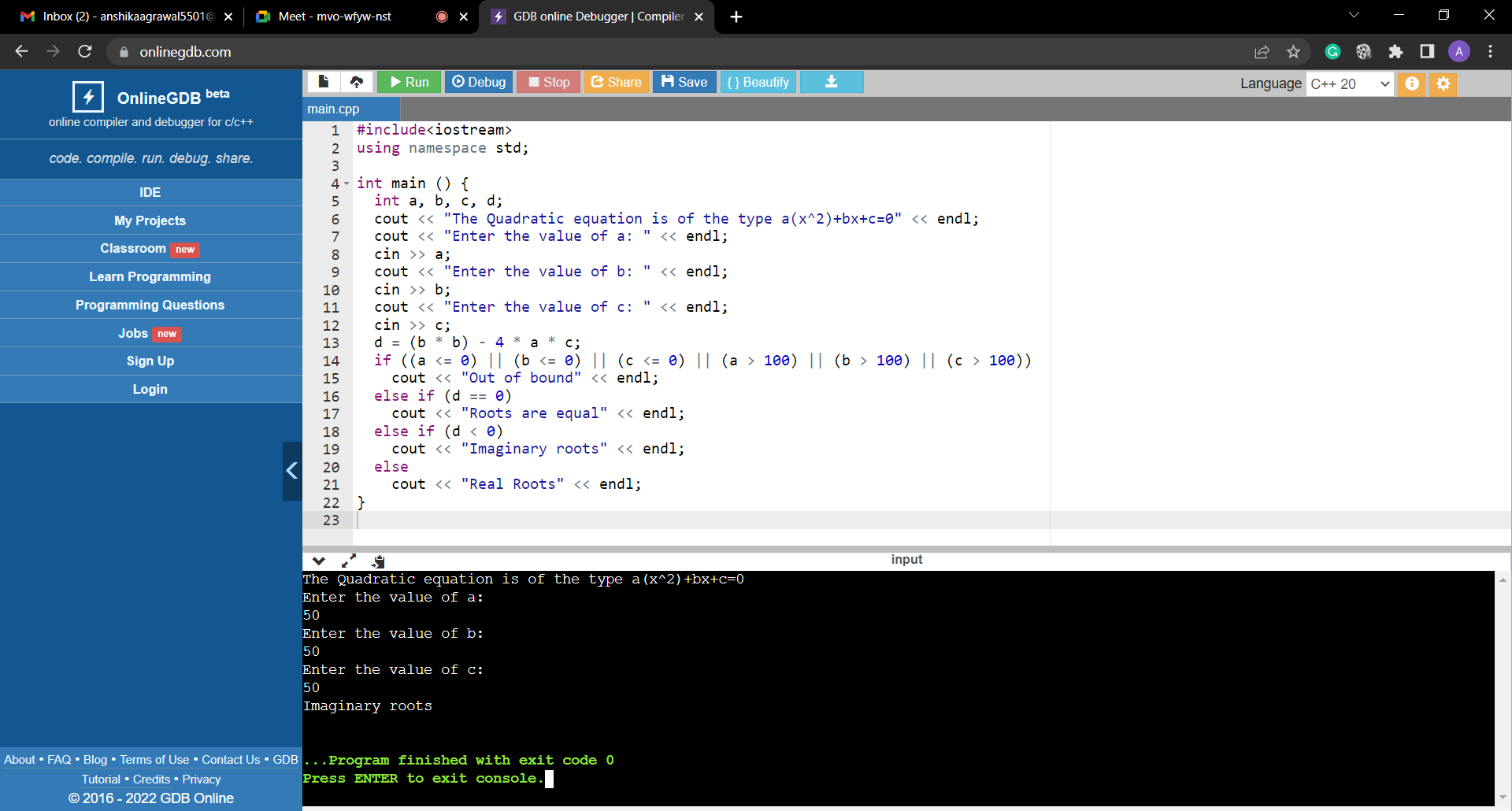
**ROBUST CASES:**

Here, we go outside the legitimate boundary, it is an extension of boundary value analysis. total test cases,= 6\*n+1 = 6\*3+1 = 19

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Testcase** | **a** | **b** | **c** | **Expected**  **Output** | **Observed**  **Output** |
| 1 | 0 | 50 | 50 | Not a Quadratic Equation | Not a Quadratic Equation |
| 2 | 1 | 50 | 50 | Real Roots | Real Roots |
| 3 | 2 | 50 | 50 | Real Roots | Real Roots |
| 4 | 50 | 50 | 50 | Imaginary Roots | Imaginary Roots |
| 5 | 99 | 50 | 50 | Imaginary Roots | Imaginary Roots |
| 6 | 100 | 50 | 50 | Imaginary Roots | Imaginary Roots |
| 7 | 101 | 50 | 50 | Out of bound | Out of bound |
| 8 | 50 | 0 | 50 | Out of bound | Out of bound |
| 9 | 50 | 1 | 50 | Imaginary Roots | Imaginary Roots |
| 10 | 50 | 2 | 50 | Imaginary Roots | Imaginary Roots |
| 11 | 50 | 99 | 50 | Imaginary Roots | Imaginary Roots |
| 12 | 50 | 100 | 50 | Equal Roots | Equal Roots |
| 13 | 50 | 101 | 50 | Out of bound | Out of bound |
| 14 | 50 | 50 | 0 | Out of bound | Out of bound |
| 15 | 50 | 50 | 1 | Real Roots | Real Roots |
| 16 | 50 | 50 | 2 | Real Roots | Real Roots |
| 17 | 50 | 50 | 99 | Imaginary Roots | Imaginary Roots |
| 18 | 50 | 50 | 100 | Imaginary Roots | Imaginary Roots |
| 19 | 50 | 50 | 101 | Out of bound | Out of bound |

**Output:**

**Viva Questions:**

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

**Ans.** When we have some coding error, it leads a program to its breakdown, which is known as a bug. The Problem in code leads to errors, which means that a mistake can occur due to the developer's coding error as the developer misunderstood the requirement or the requirement was not defined correctly. The developers use the term error. The Defect is the difference between the actual outcomes and expected outputs.

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

**Ans.** Boundary Value Analysis is based on testing the boundary values of valid and invalid partitions. The behaviour at the edge of the equivalence partition is more likely to be incorrect than the behaviour within the partition, so boundaries are an area where testing is likely to yield defects.

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

**Ans.** Black box testing involves testing a system with no prior knowledge of its internal workings. A tester provides an input, and observes the output generated by the system under test.

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

**Ans.** To apply boundary value testing, the analysis is done on the boundaries, taking the extreme ends. The maximum value is 150 and the minimum value is 1. The invalid values in this test case will be 0 and 151. Therefore, there will be four boundary value tests for such a scenario.

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

**Ans.** The disadvantages of the BVA method are as follows. It cannot test all test case input values. BVA technique is not suitable with Boolean Variable.

**Q6. What are the limitations of boundary value analysis.**

**Ans.** The success of the testing using this technique depends on the equivalence classes identified, which further depends on the expertise of the tester and his knowledge of the application. Hence, incorrect identification of equivalence classes leads to incorrect boundary value testing.

**Experiment – 2 & 4**

**Aim:**

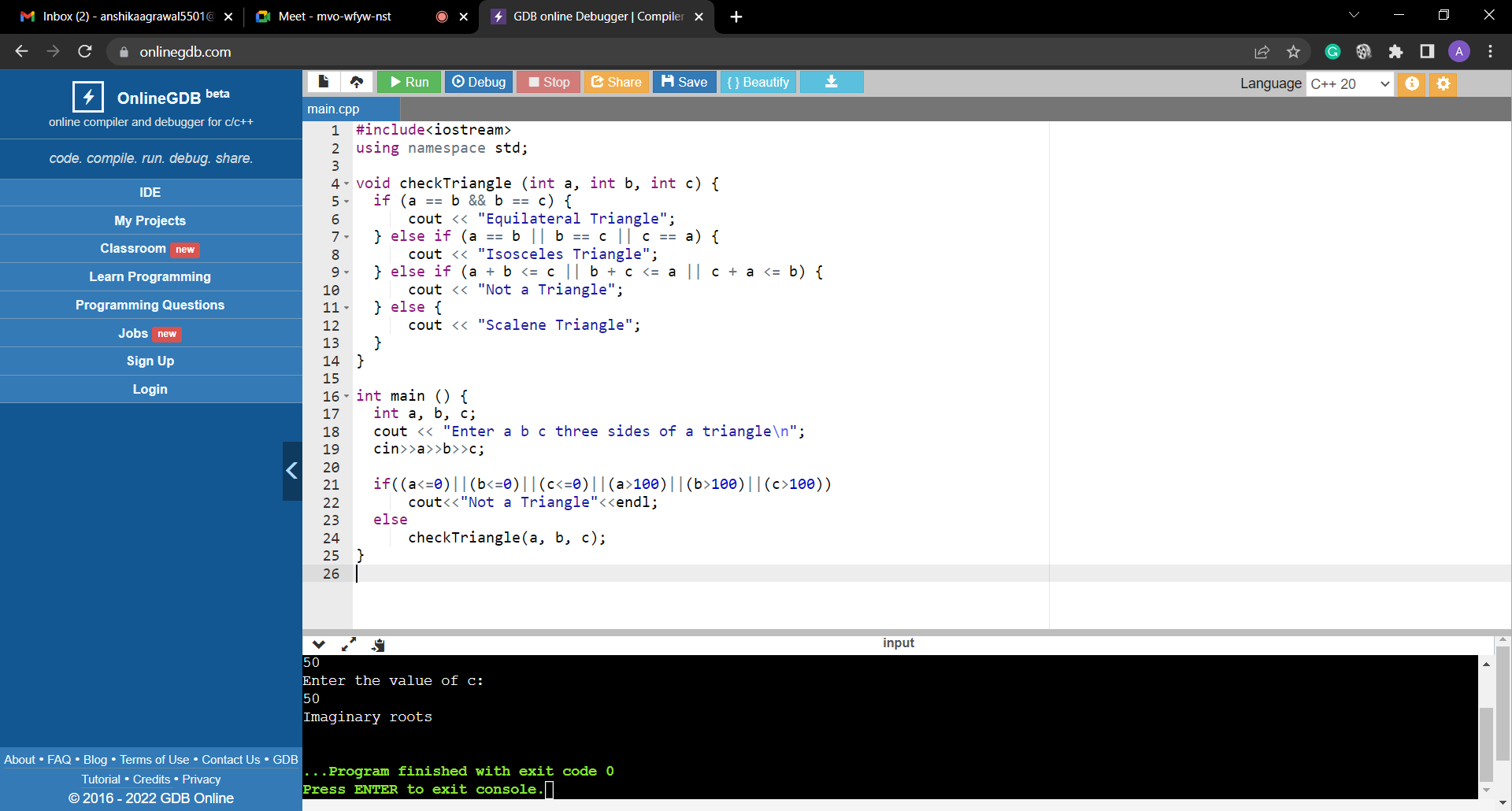
* 1. To determine the type of triangle. Its input is triple of +ve integers (say a,b,c) and the values may be from interval[1,100].The program output may be one of the following [Scalene, Isosceles, Equilateral, Not a Triangle].Perform BVA.
  2. Perform robust case testing on the above problem.

**Theory:**

**Robustness Testing**

1. 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
2. Extension of boundary value analysis
3. 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.

**Code:**



In the above program consider the values 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 is 3.

**TEST CASES:**

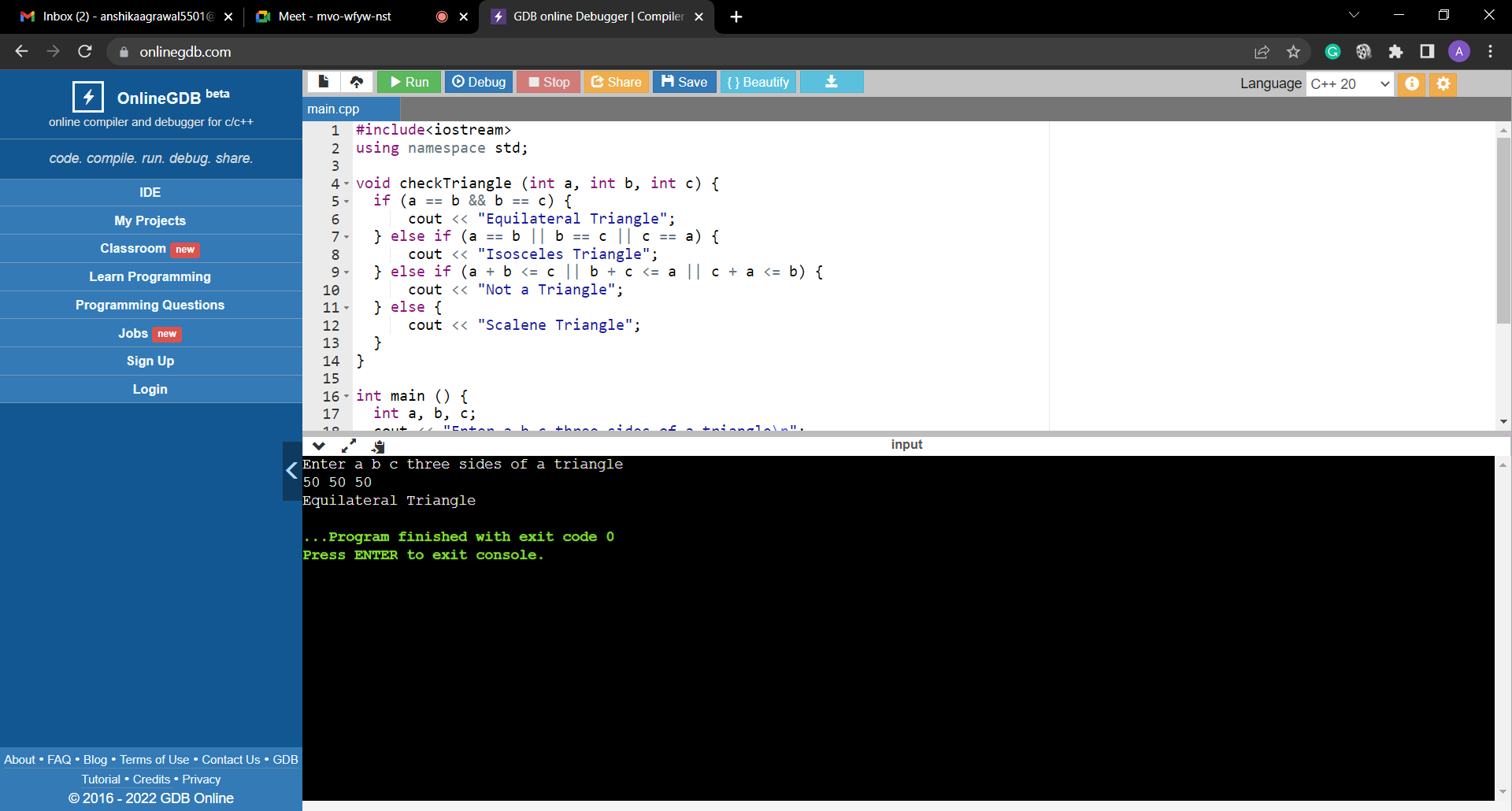
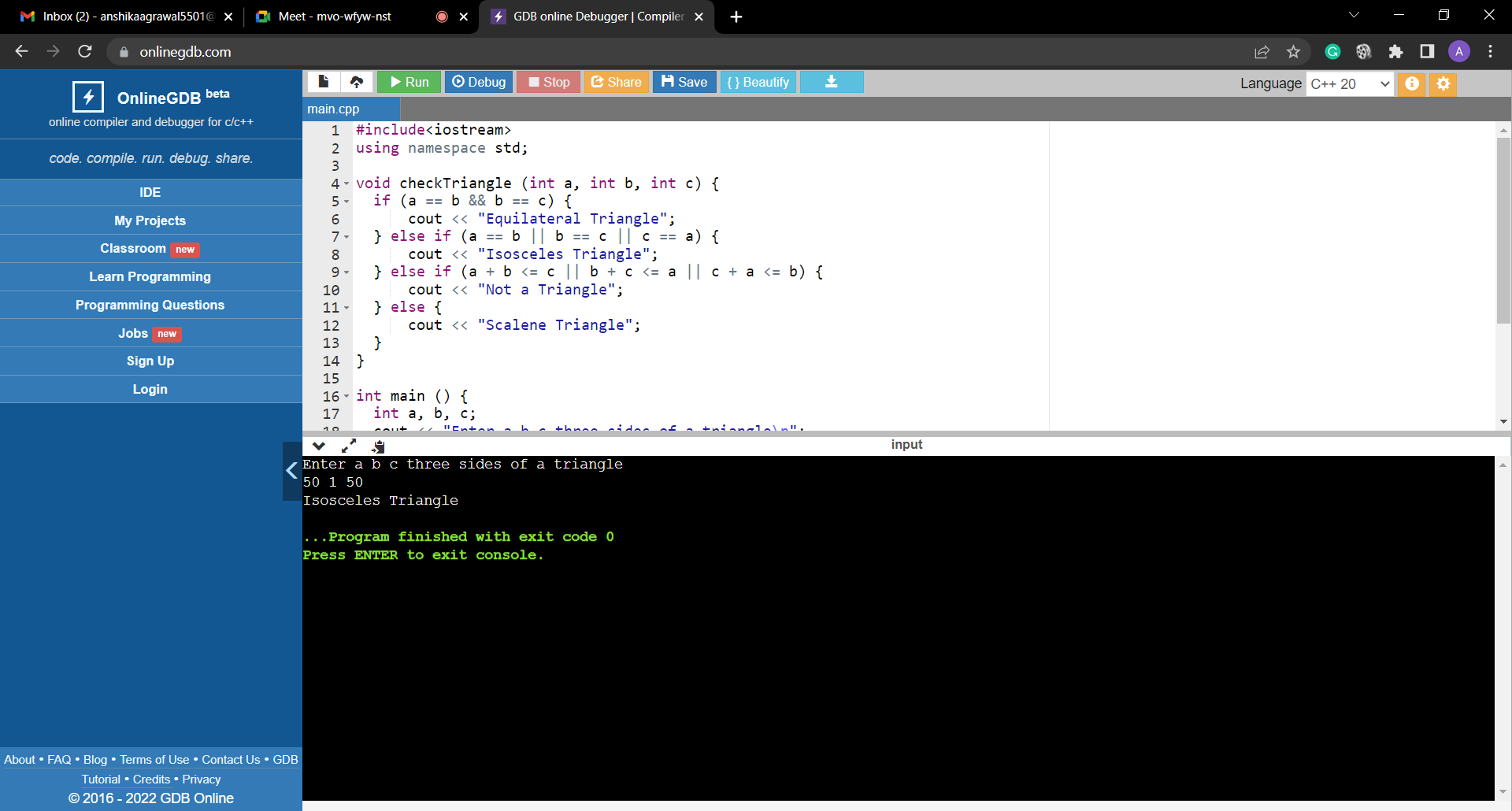
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Testcase** | **a** | **b** | **c** | **Expected**  **Output** | **Observed**  **Output** |
| 1 | 1 | 50 | 50 | Isosceles | Isosceles |
| 2 | 2 | 50 | 50 | Isosceles | Isosceles |
| 3 | 50 | 50 | 50 | Equilateral | Equilateral |
| 4 | 99 | 50 | 50 | Isosceles | Isosceles |
| 5 | 100 | 50 | 50 | Not a triangle | Not a triangle |
| 6 | 50 | 1 | 50 | Isosceles | Isosceles |
| 7 | 50 | 2 | 50 | Isosceles | Isosceles |
| 8 | 50 | 99 | 50 | Isosceles | Isosceles |
| 9 | 50 | 100 | 50 | Not a triangle | Not a triangle |
| 10 | 50 | 50 | 1 | Isosceles | Isosceles |
| 11 | 50 | 50 | 2 | Isosceles | Isosceles |
| 12 | 50 | 50 | 99 | Isosceles | Isosceles |
| 13 | 50 | 50 | 100 | Not a triangle | Not a triangle |

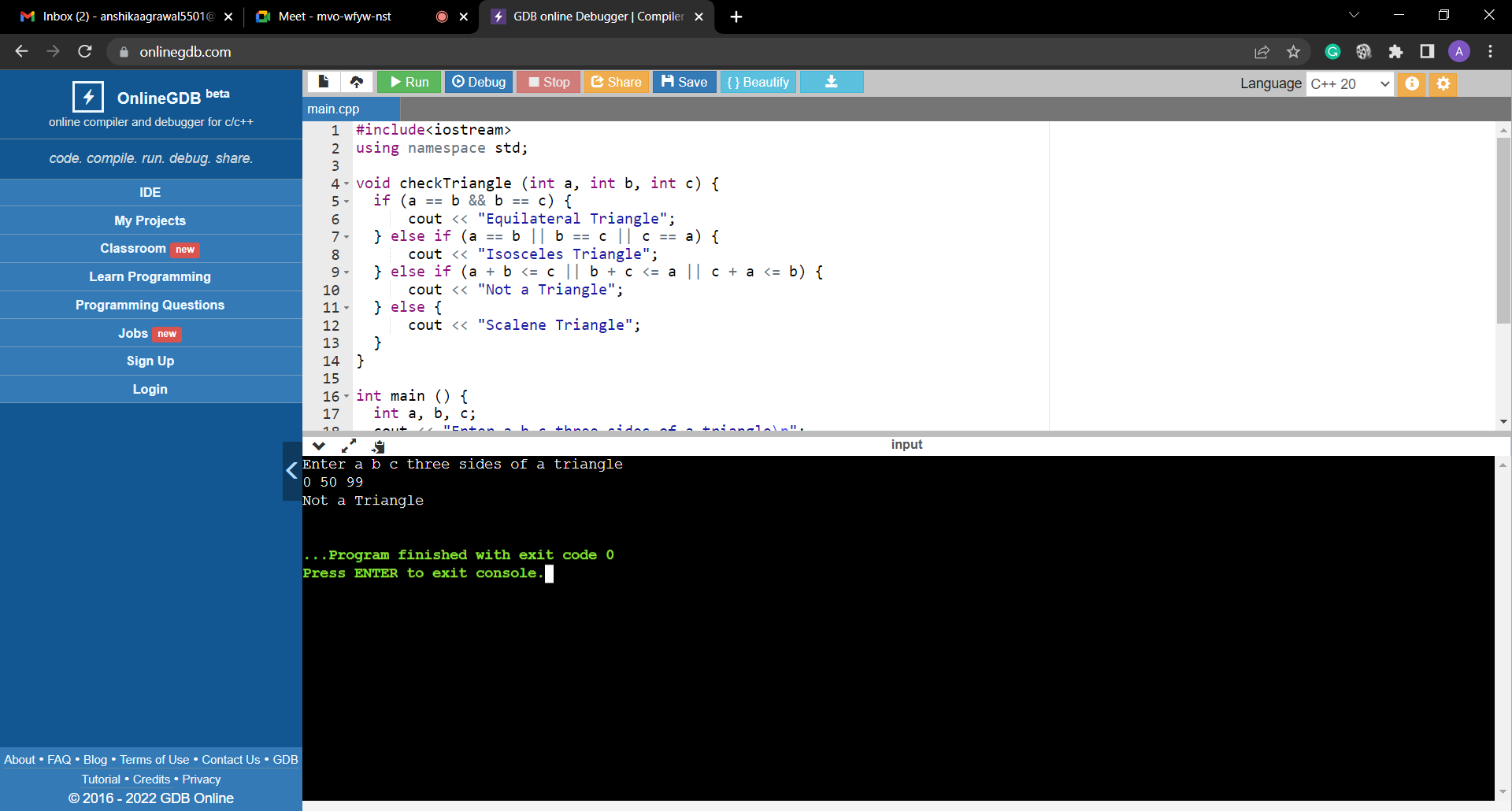
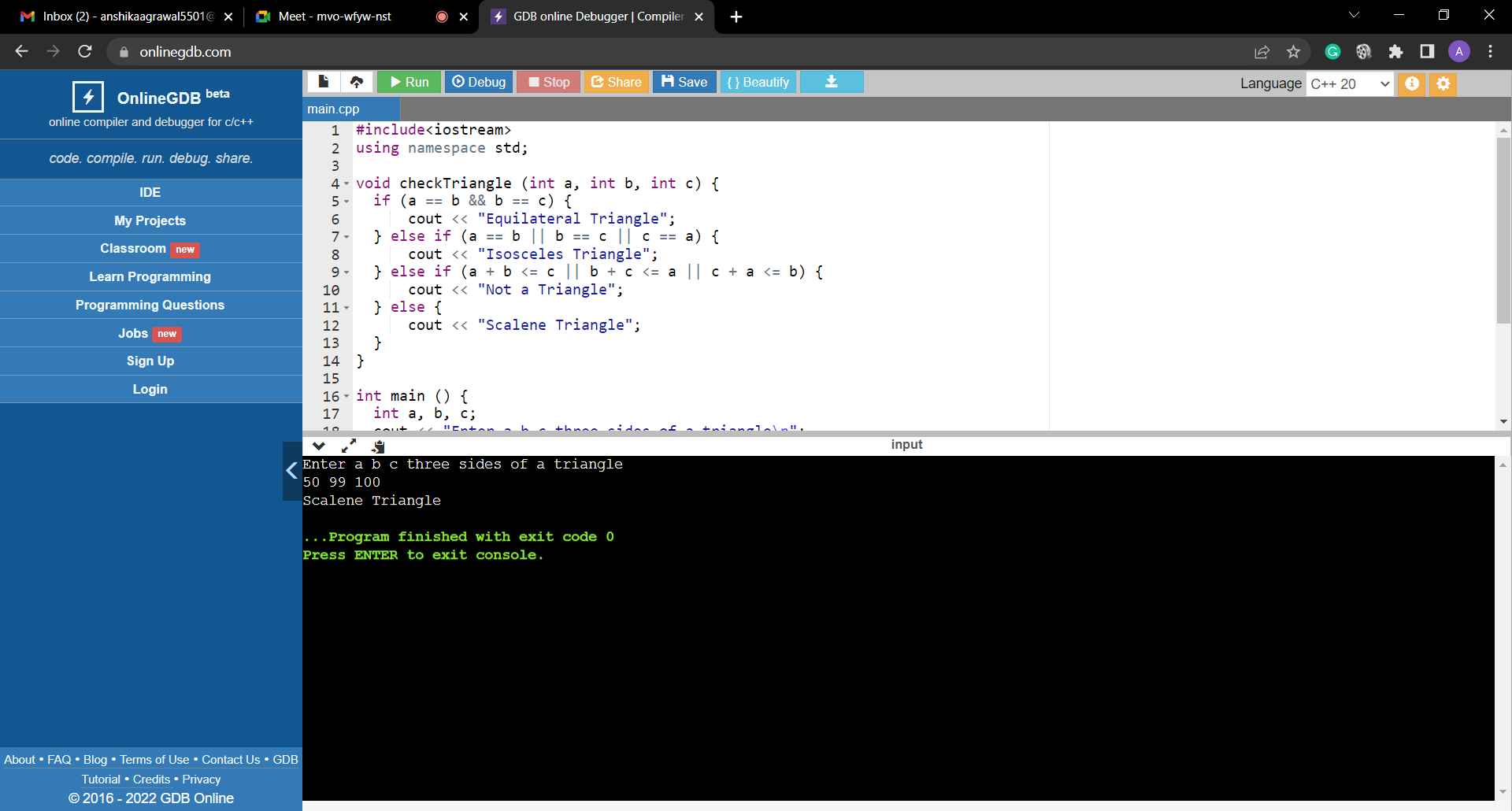
**ROBUST CASES**:

**C**onsider the values 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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Testcase** | **a** | **b** | **c** | **Expected**  **Output** | **Observed**  **Output** |
| 1 | 0 | 50 | 50 | Not a triangle | Not a triangle |
| 2 | 1 | 50 | 50 | Isosceles | Isosceles |
| 3 | 2 | 50 | 50 | Isosceles | Isosceles |
| 4 | 50 | 50 | 50 | Equilateral | Equilateral |
| 5 | 99 | 50 | 50 | Isosceles | Isosceles |
| 6 | 100 | 50 | 50 | Not a triangle | Not a triangle |
| 7 | 101 | 50 | 50 | Not a triangle | Not a triangle |
| 8 | 50 | 0 | 50 | Not a triangle | Not a triangle |
| 9 | 50 | 1 | 50 | Isosceles | Isosceles |
| 10 | 50 | 2 | 50 | Isosceles | Isosceles |
| 11 | 50 | 99 | 50 | Isosceles | Isosceles |
| 12 | 50 | 100 | 50 | Not a triangle | Not a triangle |
| 13 | 50 | 101 | 50 | Not a triangle | Not a triangle |
| 14 | 50 | 50 | 0 | Not a triangle | Not a triangle |
| 15 | 50 | 50 | 1 | Isosceles | Isosceles |
| 16 | 50 | 50 | 2 | Isosceles | Isosceles |
| 17 | 50 | 50 | 99 | Isosceles | Isosceles |
| 18 | 50 | 50 | 100 | Not a triangle | Not a triangle |
| 19 | 50 | 50 | 101 | Not a triangle | Not a triangle |

**Output:**



**Viva Questions:**

**Q1. What is equivalence partitioning explain with example.**

**Ans.** Equivalence Partitioning Method is also known as Equivalence class partitioning (ECP). It is a software testing technique or black-box testing that divides input domain into classes of data, and with the help of these classes of data, test cases can be derived.

Example: Let us consider an example of any college admission process. There is a college that gives admissions to students based upon their percentage.

Consider percentage field that will accept percentage only between 50 to 90 %, more and even less than not be accepted, and application will redirect user to an error page. If percentage entered by user is less than 50 %or more than 90 %, that equivalence partitioning method will show an invalid percentage. If percentage entered is between 50 to 90 %, then equivalence partitioning method will show valid percentage.

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

**Ans.** The test execution environment configured for testing. Test bed consists of specific hardware, software, Operating system, network configuration, the product under test, other system software and application software.

Test data is data which has been specifically identified for use in tests, typically of a computer program.

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

**Ans.** Bugs in software can arise from mistakes and errors made in interpreting and extracting users' requirements, planning a program's design, writing its source code, and from interaction with humans, hardware and programs, such as operating systems or libraries.

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

**Ans.** There is no written rule. According to BCS/ISTQB Software Testing Foundation, you cannot physically test for every scenario. When deciding how much testing you should carry out, you may want to consider the level of risk involved, including technical and business risk and even budget or time constraints.

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

**Ans.** Verification is a process of determining if the software is designed and developed as per the specified requirements. Validation is the process of checking if the software (end product) has met the client's true needs and expectations.

**Experiment – 5**

**Aim:** Create a test plan document for any application (e.g. 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 texting

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. These 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

**Viva Questions:**

**Q1. Why should we test?**

**Ans.** The basic goal of testing is to uncover as many flaws as possible in a software product while confirming whether or not the application meets the user's needs. Defects should be found as early in the testing cycle as feasible.

**Q2. What is test Suite?**

**Ans.** Test suite is a container that has a set of tests which helps testers in executing and reporting the test execution status. It can take any of the three states namely Active, In progress and completed.

**Q3. Define Test, Test Case.**

**Ans.** Test is the process of evaluating and verifying that a software product or application does what it is supposed to do.

The test case is defined as a group of conditions under which a tester determines whether a software application is working as per the customer's requirements or not.

**Q4. How much testing is enough?**

**Ans.** There is no written rule. According to BCS/ISTQB Software Testing Foundation, you cannot physically test for every scenario. When deciding how much testing you should carry out, you may want to consider the level of risk involved, including technical and business risk and even budget or time constraints.

**Q5. What is Test plan?**

**Ans.** The test plan conveys how the test will be performed. This includes defining test objectives, test approach, test tools, test environment, test schedules and team responsibilities and composition.

**Experiment – 6**

**Aim:** Study of Any Testing Tool (Win Runner).

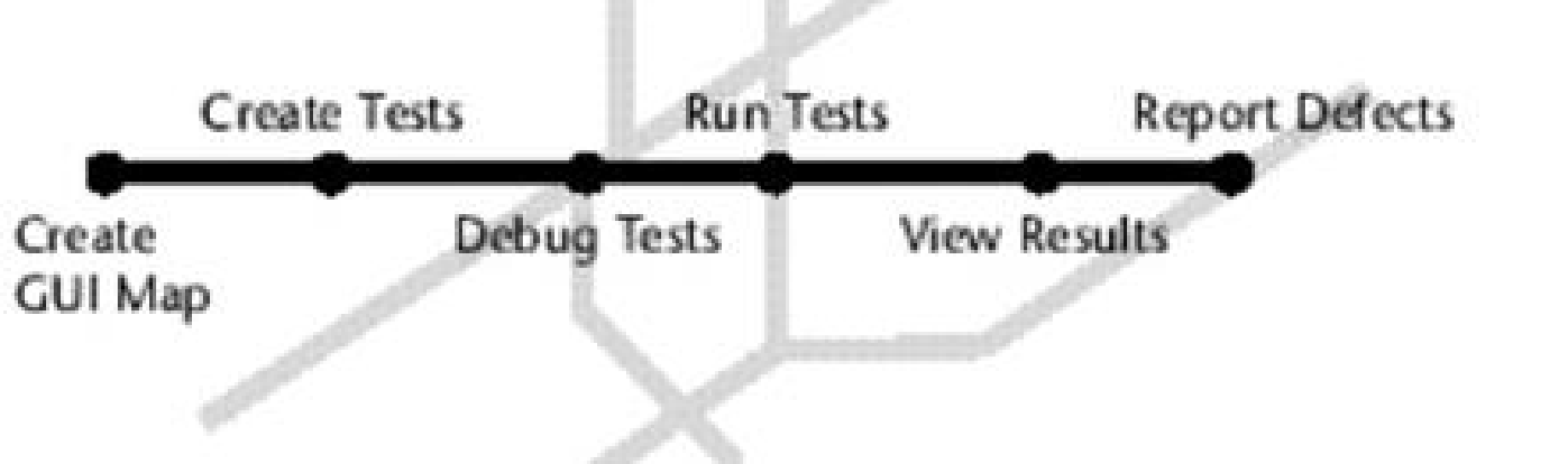
**Theory:**

**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

**Win Runner 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: - Win runner 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, win runner 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

**Win Runner environment**

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

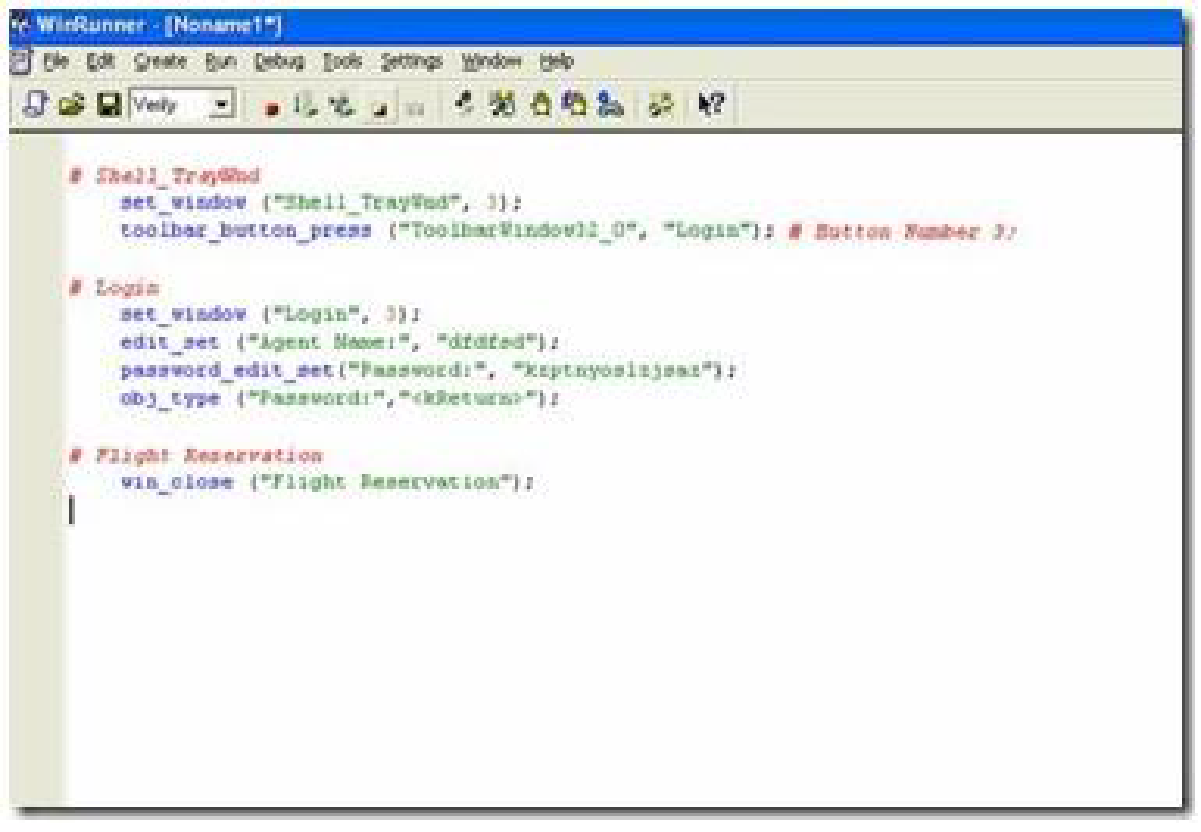
**Exploring the WinRunner Window**

**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.

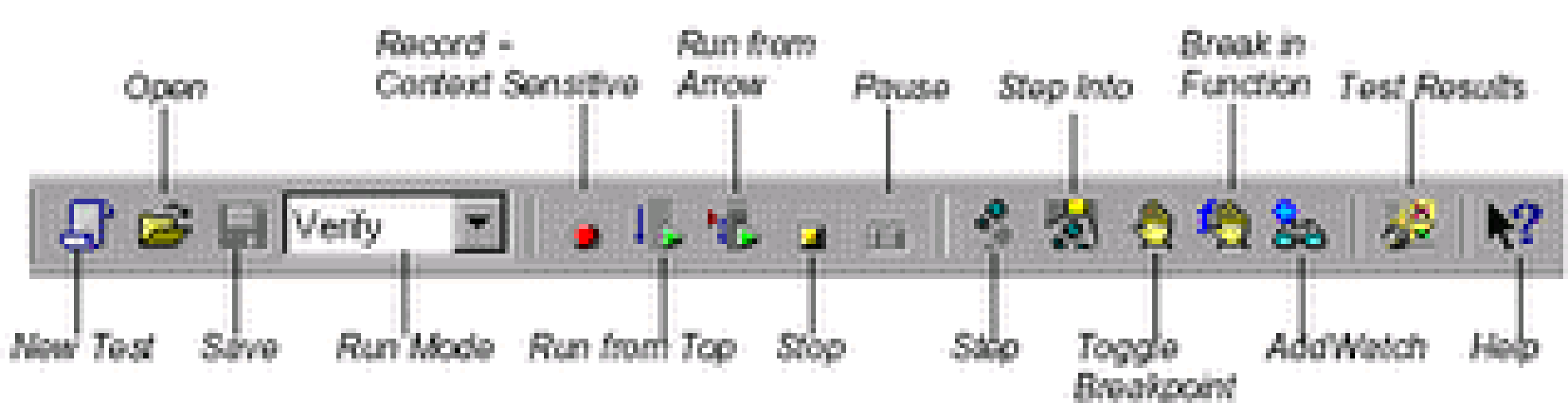
****

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 the WinRunner tool and report the defects?**

**Ans.** When you finish any test in WinRunner, it displays the results in a report format. The report logs general information about the test run i.e. date, operator mode and total run time. Also, the report details include all the major events that occurred during the run, such as checkpoints, error messages, system messages, or user messages. A mismatch can be found in the report panel by comparing the Actual result and the Expected result. If a test run fails due to a defect in the application being tested, you 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.

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

**Ans.** There are two types of Recording in WinRunner:

1. Context-Sensitive Recording, which records the operations you perform on your application by identifying Graphical User Interface (GUI) objects. WinRunner identifies all the objects in your window that you have clicked on like menus, windows, lists, buttons and the type of operation you perform such as enable, move, select, etc.
2. Analog Recording records keyboard input, mouse clicks, and the precise x- and y-coordinates travelled by mouse pointer across the screen i.e WinRunner records exact coordinates travelled by mouse.

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

**Ans.** WinRunner fails to identify an object in a GUI due to various reasons. A few of them are enlisted

below.

1. The object is not a standard windows object.

2. If the browser used is not compatible with the WinRunner version, then the GUI Map

Editor will not be able to learn any of the objects displayed in the browser window

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

**Ans.** When you click an object, WinRunner assigns the object a Logical name, which is usually the object’s text label. Logical names make it easy for you to read the test script.

For example, when you have selected the Order No. checkbox, WinRunner has recorded the following statement in WinRunner TSL:

button\_set (“Order No.”, ON);

“Order No.” is the object’s logical name.

An object’s Logical name is determined by its class. In most cases, the logical name is the label

that appears on an object.

**Q5. What is the purpose of different record methods:**

1. **Record**

Record instructs WinRunner to record all operations performed on a GUI object. This is

the default record method for all classes. (The only exception is the static class (static

text), for which the default is Pass Up.)

1. **Pass up**

Pass Up instructs WinRunner to record an operation performed in this class as an

operation performed on the element containing the object. Usually, this element is a

window, and the operation is recorded as win\_mouse\_click.

1. **As Object**

As Object instructs WinRunner to record all operations performed on a GUI object as

though its class was the “object” class.

1. **Ignore**

Ignore instructs WinRunner to disregard all operations performed in the class.

**Experiment – 7**

**Aim:** Study of Any Test Management Tool (QA Complete)

**Theory:**

**Introduction**

QA Complete is a Test management tool that can be used for both manual and Automated

Testing. It is a tool with powerful test management capabilities that allow us to track all aspects

of software quality in an efficient manner. QA Complete supports all aspects of the 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.

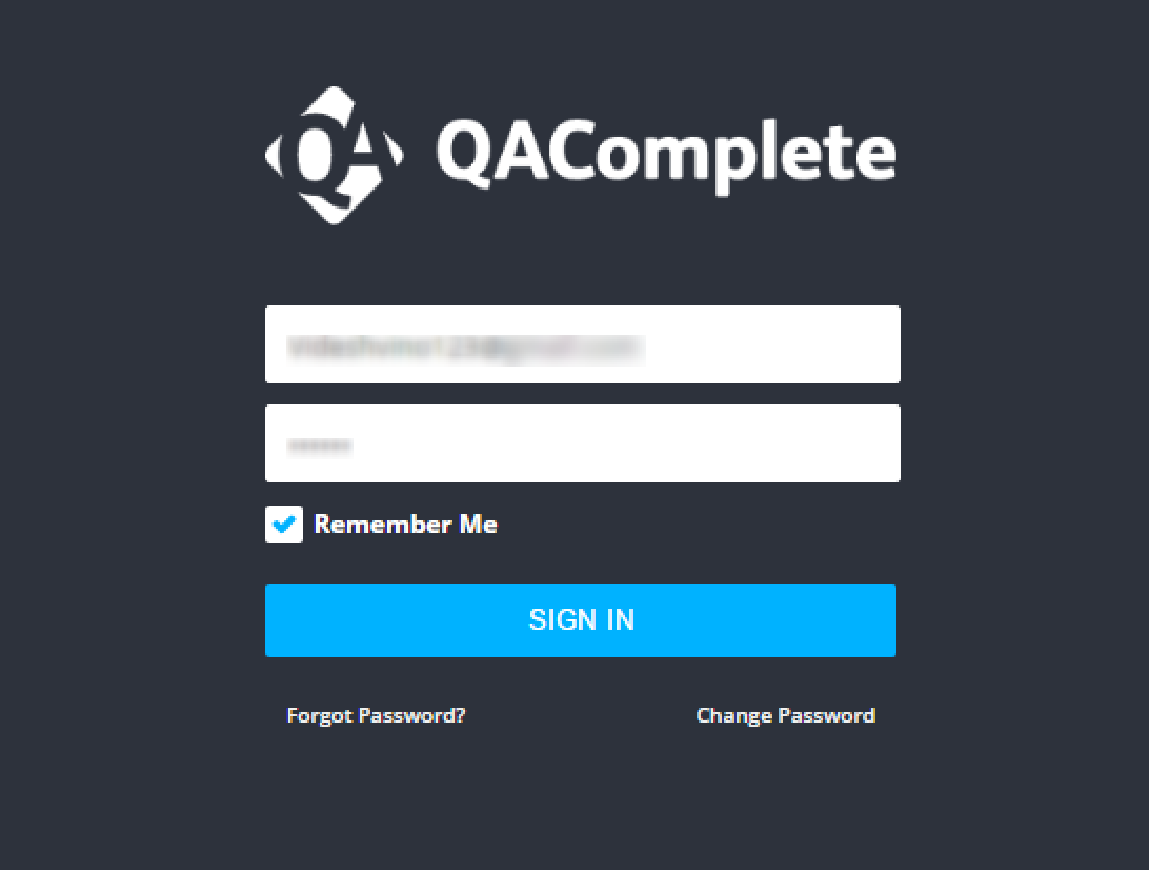
**Features**

* 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 – This can be integrated with various automation tools, and it allows us to track and report overall (manual and automated) Test Management efforts at a single place
* Bug Tracker Integration – Can be integrated with various Bug tracking tools

**Steps to Setup and Work on the QA Complete**

To manage and produce the right test deliverables, let us assume an E-Work Sight Login Page needs to be tested manually. The following steps and screenshots will explain how we can manage the Test Deliverables using QA Complete Test Management Tool.

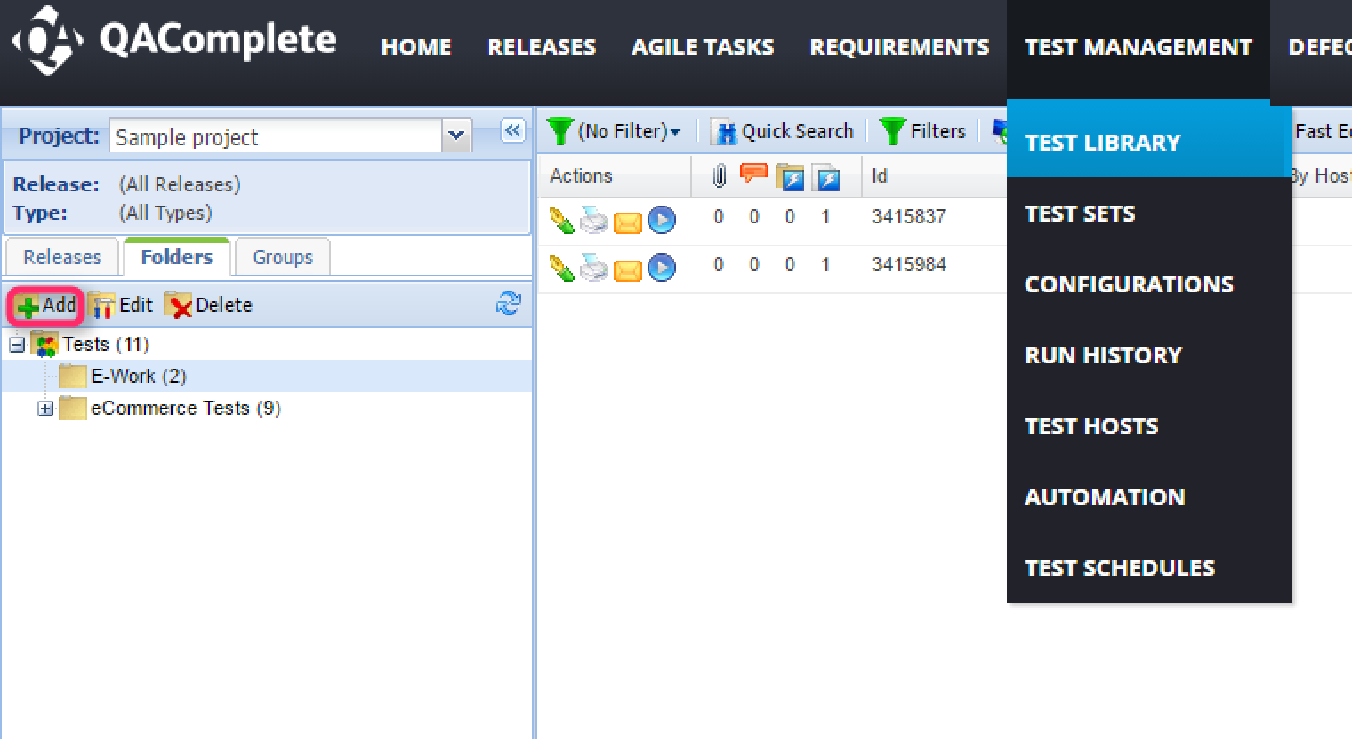
Step 1: Log into QA Complete Tool



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

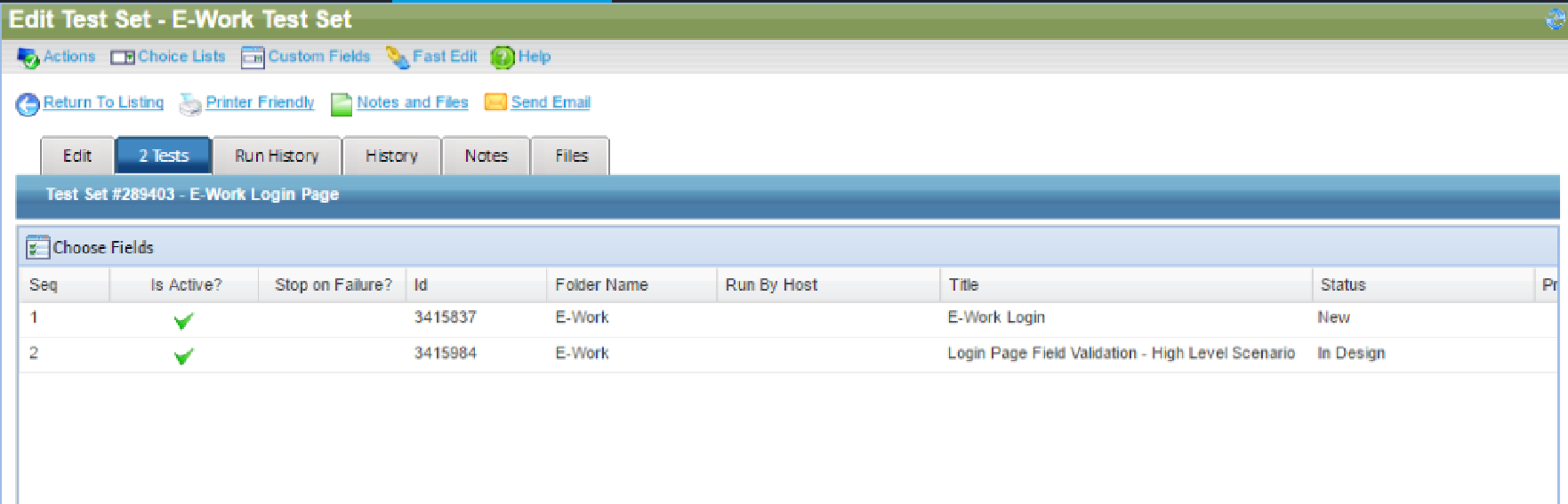


Step 3: Navigate to the Test Management Tab -> Test Library -> Add New folder

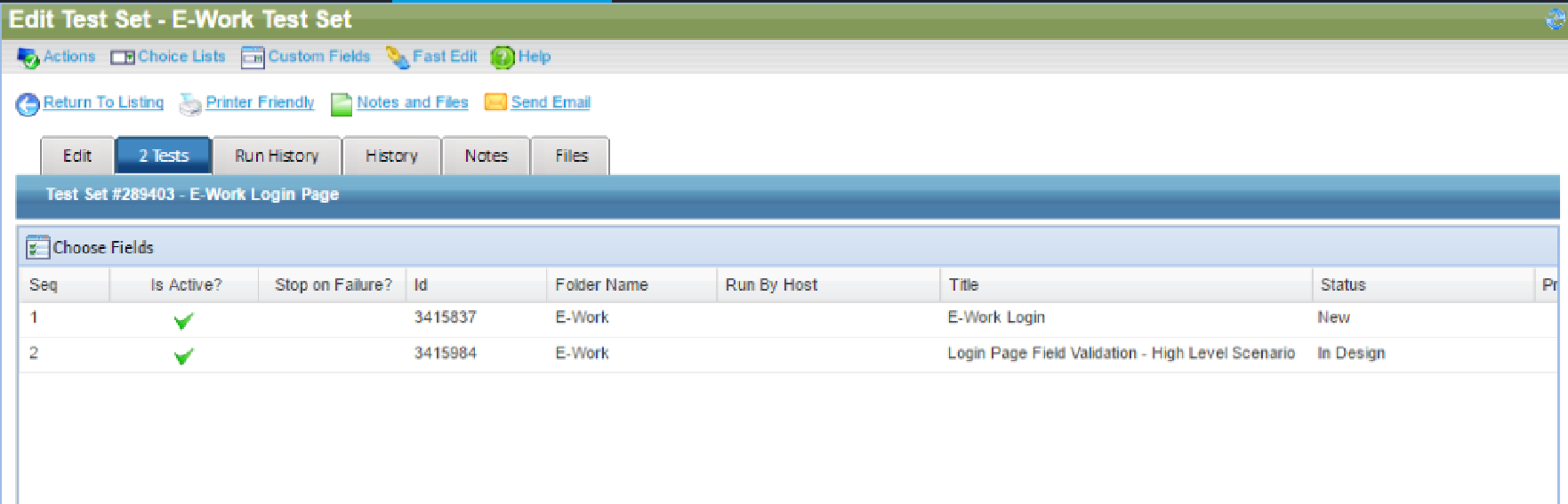


Step 4: Navigate to Test Management Tab -> Test Sets -> Create a folder (Add the ‘+’ @ left panel) -> Create a new test set using the Add New icon -> After entering the details click the Submit button -> Navigate to the Tests Tab and Design the steps accordingly











Step 5: To run the Test Sets -> Test Management Tool -> Test Sets -> Click the RUN icon



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.



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



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



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

****

**Viva Question:**

**Q1. What are bug leakage and bug release?**

**Ans.** A defect that exists during testing yet is unfound by the tester which is eventually found by the

tester/end-user is also called bug leakage.

A bug release is when a particular version of s/w is released with a set of known bug(s)/defect(s).

These bugs are usually low severity and/or low priority bugs. It is done when the company can afford the existence of a bug in the released s/w rather than the time/cost for fixing it in that particular version. These bugs are usually mentioned in the Release Notes.

**Q2. What is the difference between build and release?**

**Ans.** The main difference between Build and Release in Software Testing is that Build is a version of

software the development team hands over to the testing team for testing purposes while Release

is software the testing team hands over to the customer.

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

**Ans.** QA deals more with the management stuff: methods and techniques of development, project analysis, checklists, etc. QA goes through the whole product life cycle (SDLC) and heads the process of software maintenance.

Software Testing is a way of exploring the system to check how it operates and find the possible defects.

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

**Ans.** Test Automation Problems are listed below:

1. Unrealistic expectations of automated testing.
2. Using the wrong tools.
3. Automate useless tests and neglect important test cases.
4. Choosing the wrong testing time.
5. Lack of proper testing.

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

**Ans.** Agile testing enables collaboration and consistent communication between the development and

testing teams. As a result, complex issues are prevented or solved quickly. In addition to the strong team, the testing team can be a part of the production process instead of entering just prior to release.

Testing is a process that ensures that the product will meet all the requirements in each iteration as there is a sign of progress.

**Q6. Explain what should your QA documents include?**

**Ans.** The QA documentation is used for coordination, control, delivery, managing or support for the

item which is required for quality assurance purposes.

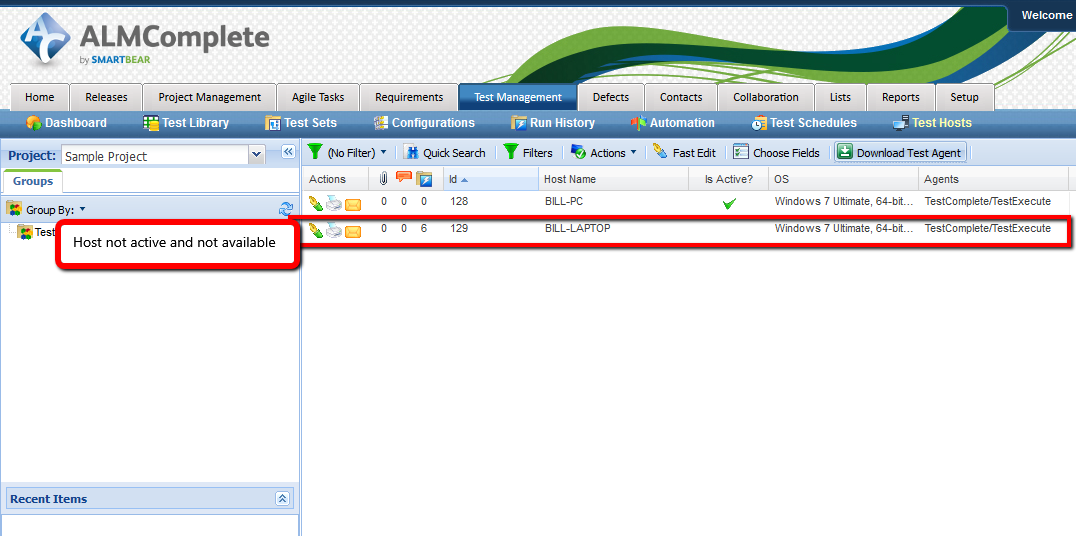
QA documents include:

* How to create the test cases and the reason for the same,
* The life cycle of a test case.
* Test plan along with test suits and checklists.

**Experiment – 8**

**Aim:** Automate the Test cases using Test Automation tool(using QA Complete)

**Setting up automated tests in QA complete**



**Step1:**

Check in QA Complete that the Test Complete 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 Test Complete machine.

**Step 2**: On the Test Complete 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 ‘Test Manager Agent’ service and start the service.

**Step 3: Creating an Automate Test**

This is a 4 stage process.

1. Package up the Test Complete project suite

2. Define the Automated Test in the QA Complete Test Library

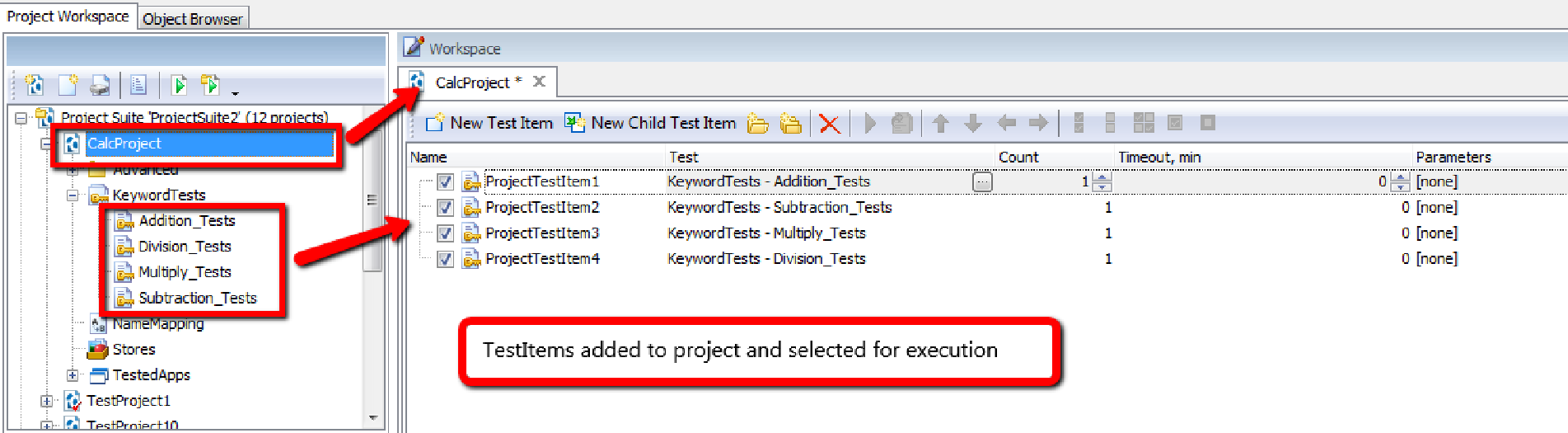
3. Execution of Automated Tests – standalone

4. Exeuction of Automated Tests – as part of a Test Set

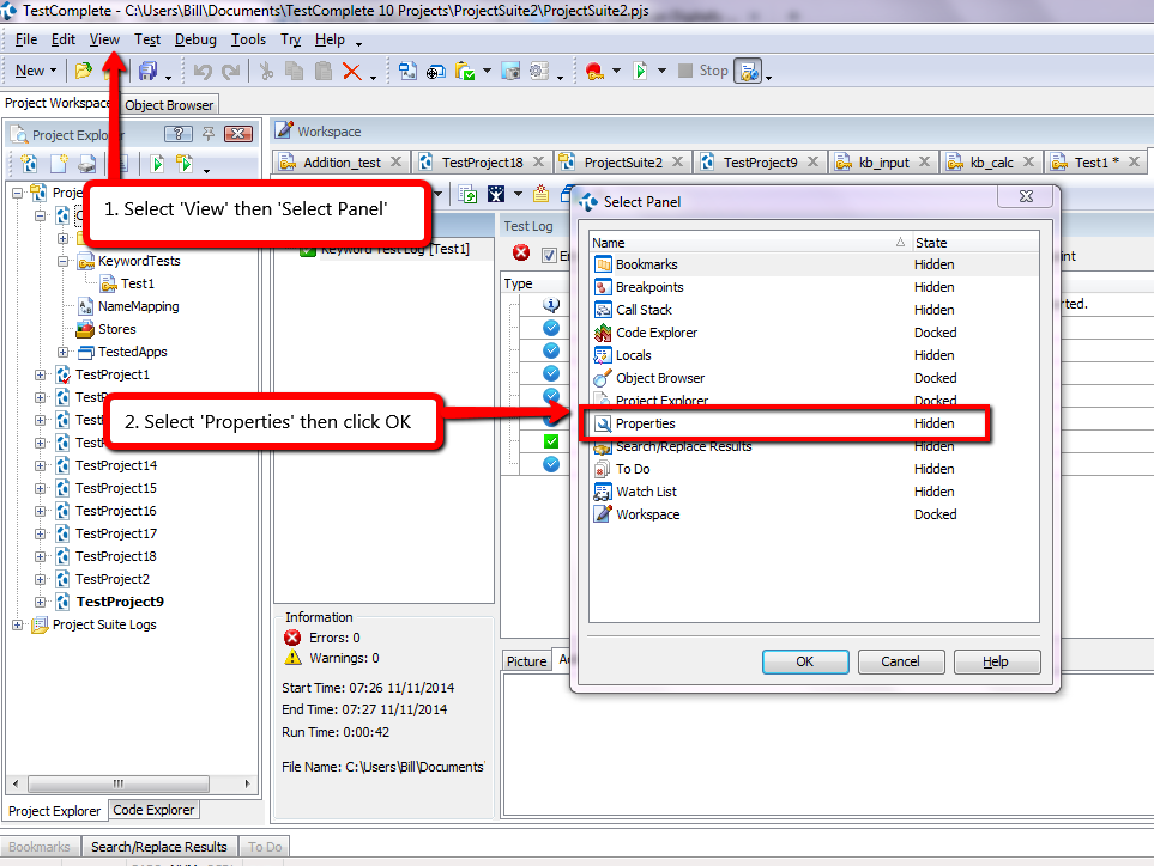
**1: Package up the Test Complete Project Suite**

To zip the project suite up follow these steps:

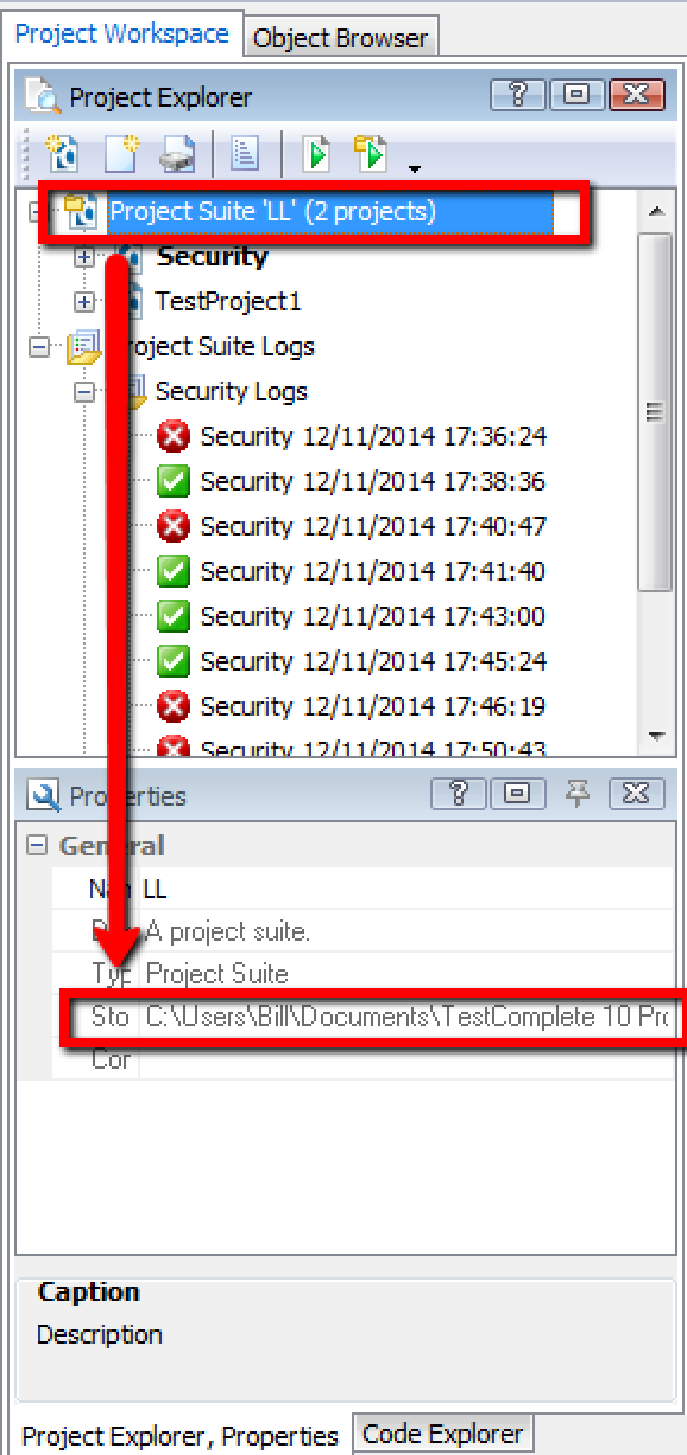
1. Make sure to define the ‘Test Items’ and enabled them within Test Complete project(s).



1. Find the location of test complete 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.



1. On the file system (or in test complete) remove the log files.
2. 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 QA Complete Test Library:**

create the test case in the ‘Test Library’ area of QA Complete and then attach the zipped up Test Complete project suite to this test case.

First we need to create a new test. Navigate to the Test Management Library area in QA Complete 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 Test Complete Project Suite.

When adding a new Test Complete project suite to QA Complete following 6 fields will be presented :

**Title:** either leave this blank and QA Complete will give this automated test the same name as the Test Complete 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.

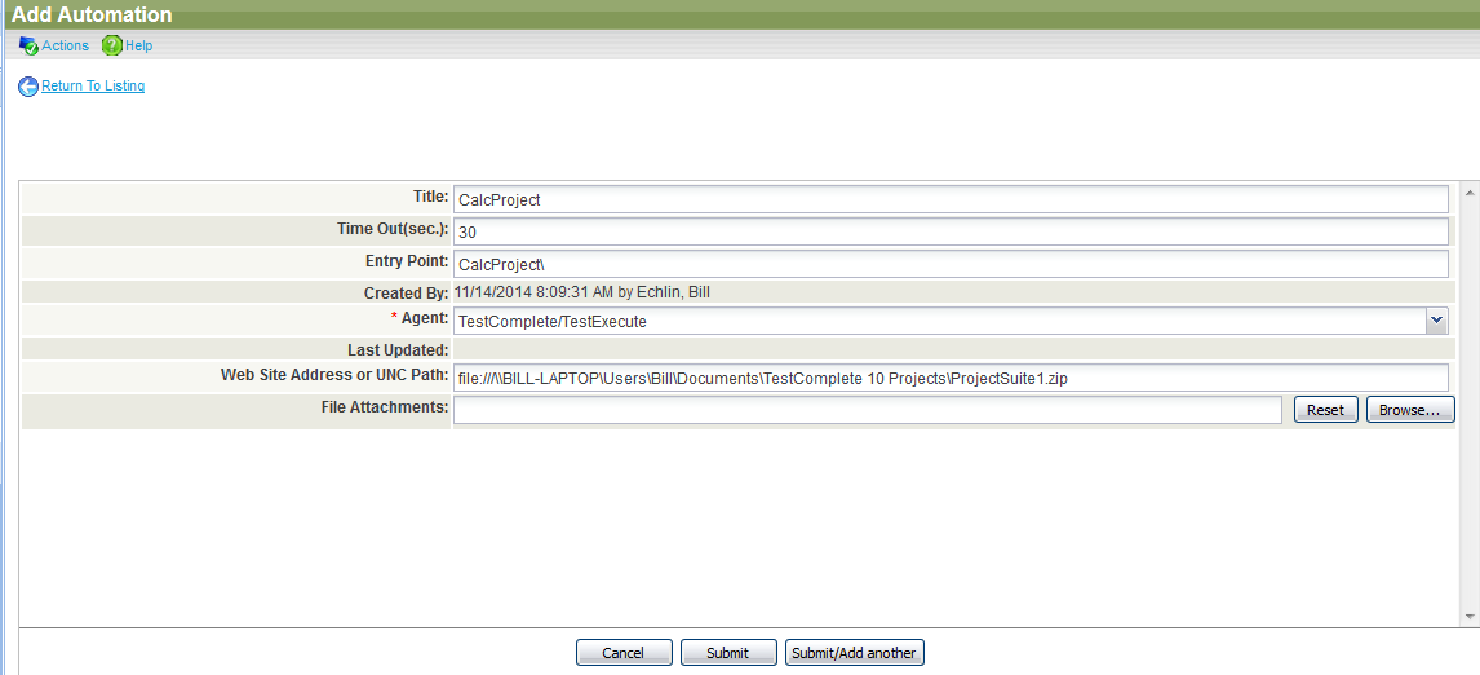
**Agent**: at the moment QA Complete only supports one type of test agent which is Test Complete/ Test Execute. Other types of test agent are in the pipeline.

**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.

**File Attachments:** attached the zipped up project suite file to the QA Complete test case and upload the file to QA Complete.

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 QA Complete automated test case.

**Viva Questions:**

**Q1. What is Automation testing?**

**Ans.** Automation Testing is a software testing technique that performs using special automated testing software tools to execute a test case suite.

**Q2. When will you automate a test?**

**Ans.** We can automate a test when we want to run the same test cases across multiple machines at the same time. Automation testing is also a good way to avoid human error in your testing.

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

**Ans.** During the planning phase of automation, the points to be considered are as follows:

1. Selection of the “right” Automation tool
2. Selection of Automation Framework if required
3. List of in scope and out of scope items for automation
4. Test Environment Setup
5. Preparing the Gantt Chart of Project timelines for test script development & execution.
6. Identify Test Deliverables

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

**Ans.** 5 Steps to get started with Automated Testing

Step 1: Defining the Scope of Automation.

Step 2: Selecting a Testing Tool.

Step 3: Planning, Designing, and Development.

Step 4: Executing Test Cases and Build your reports.

Step 5: Maintaining previous test cases.

**Q5. When will you not automate testing?**

**Ans.** If a test needs to be manually “eyeballed” to determine whether the results are correct. Test that cannot be 100% automated should not be automated at all — unless doing so will save a considerable amount of time. Test that adds no value. Test that doesn't focus on the risk areas of your application.

**Experiment – 9**

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

**Theory:**

**Bugzilla**

Bugzilla is an open-source issue/bug tracking system that allows developers to keep track of

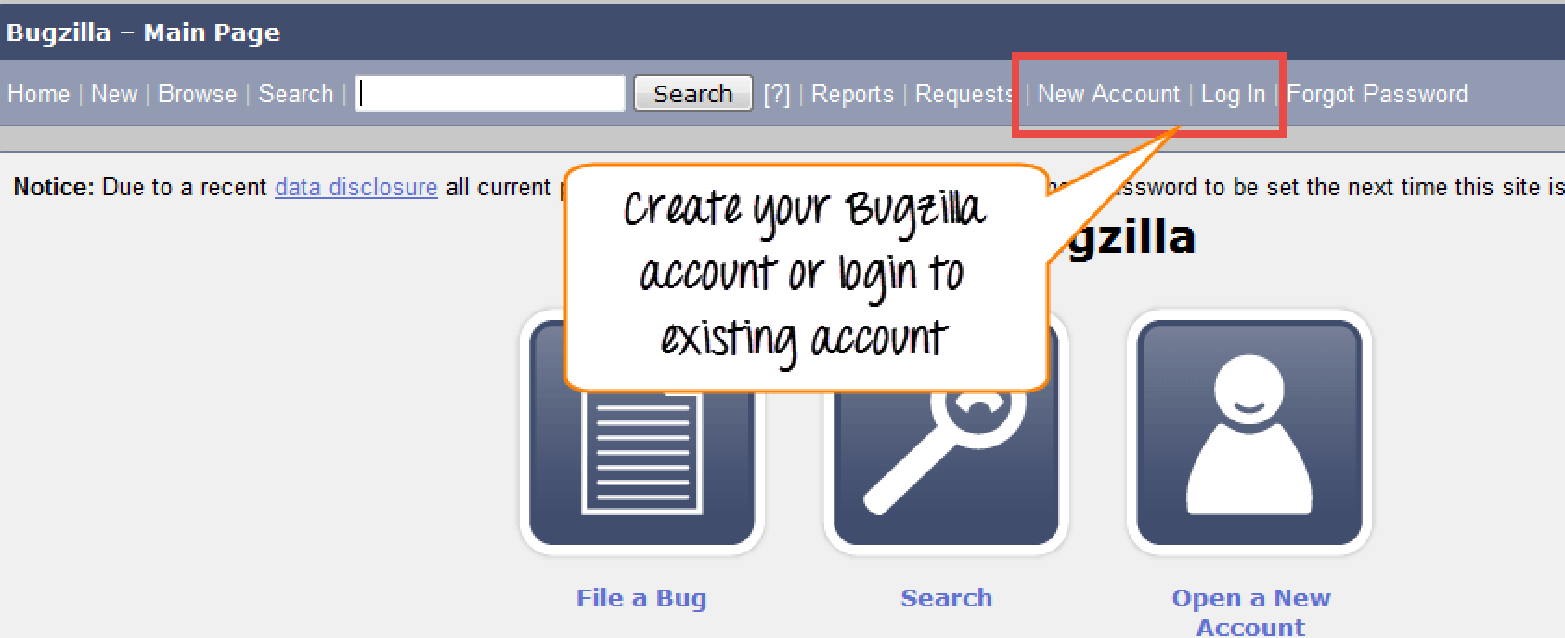
outstanding problems with their products. It is written in Perl and uses the MYSQL database. Bugzilla is a Defect tracking tool, however, it can be used as a test management tool as such it can be easily linked with other Test Case management tools like Quality Center, Testlink etc. This open bug-tracker 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

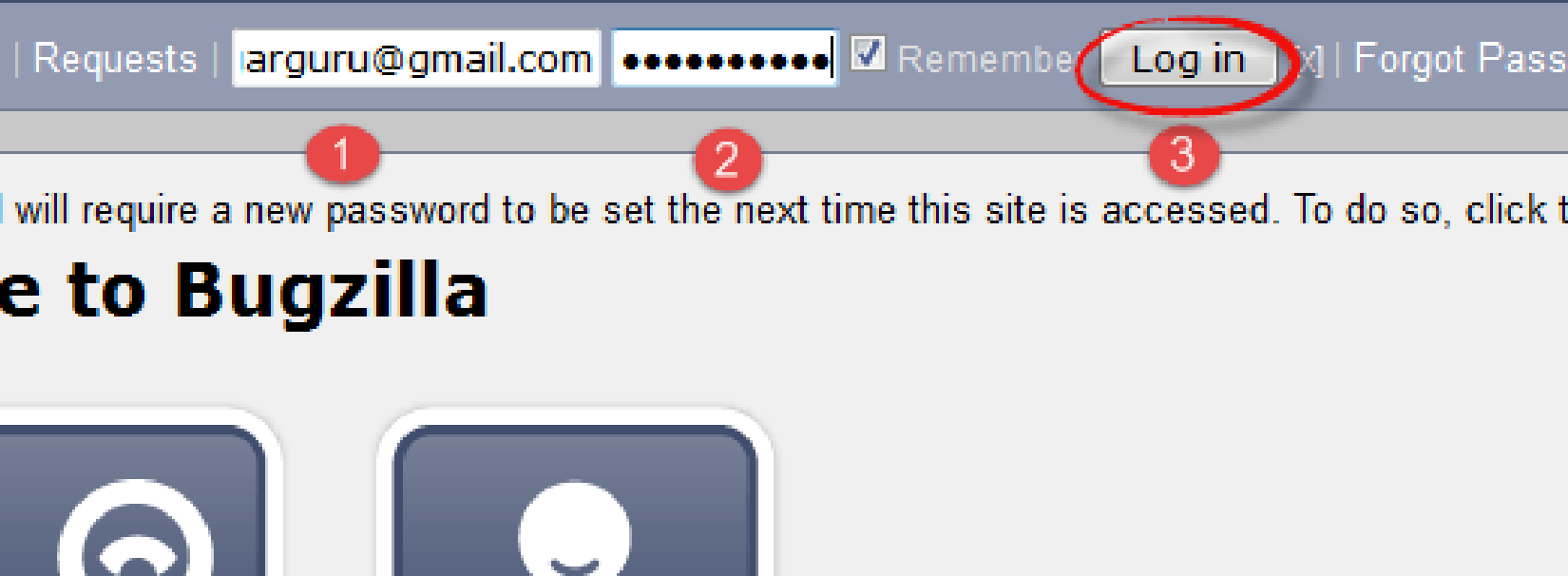
1. Advanced search capabilities
2. E-mail Notifications
3. Modify/file Bugs by e-mail
4. Time tracking
5. Strong security
6. Customization
7. Localization

**How to log in to Bugzilla**

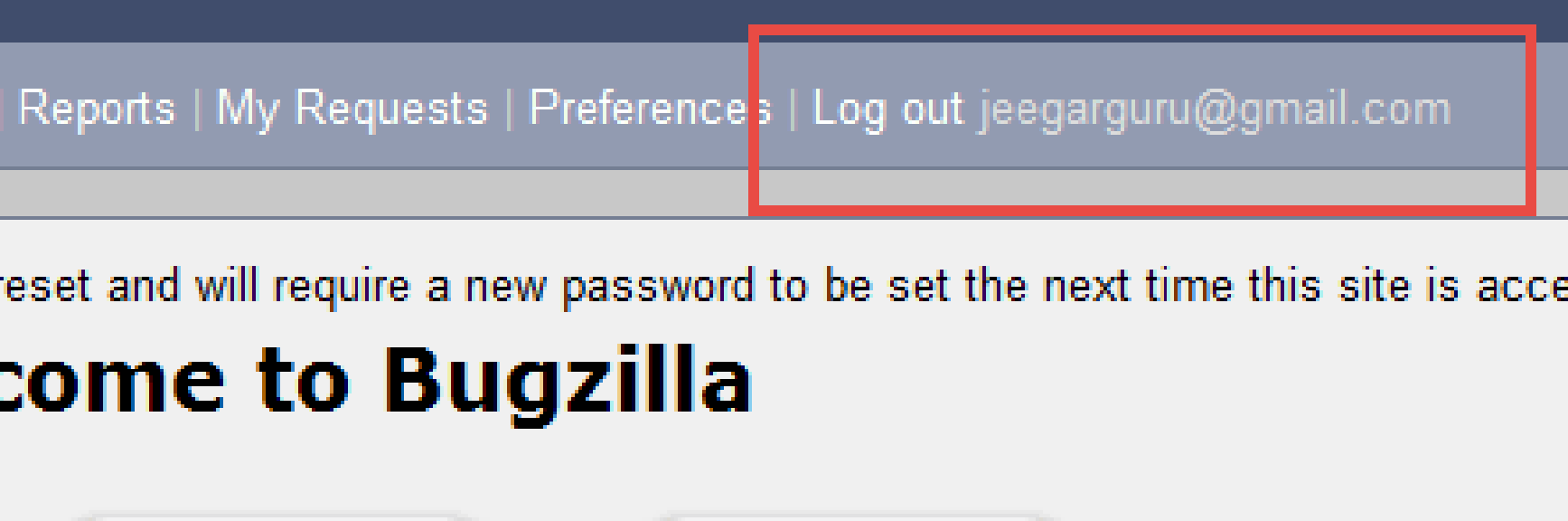
Step 1) Use the following link for your handons. To create an account in the Bugzilla tool or to login into the existing account go to New Account or Login option in the main menu.

****

Step 2) Now, enter your personal details to log into Bugzilla – User ID Password. And then click on “Log in”.

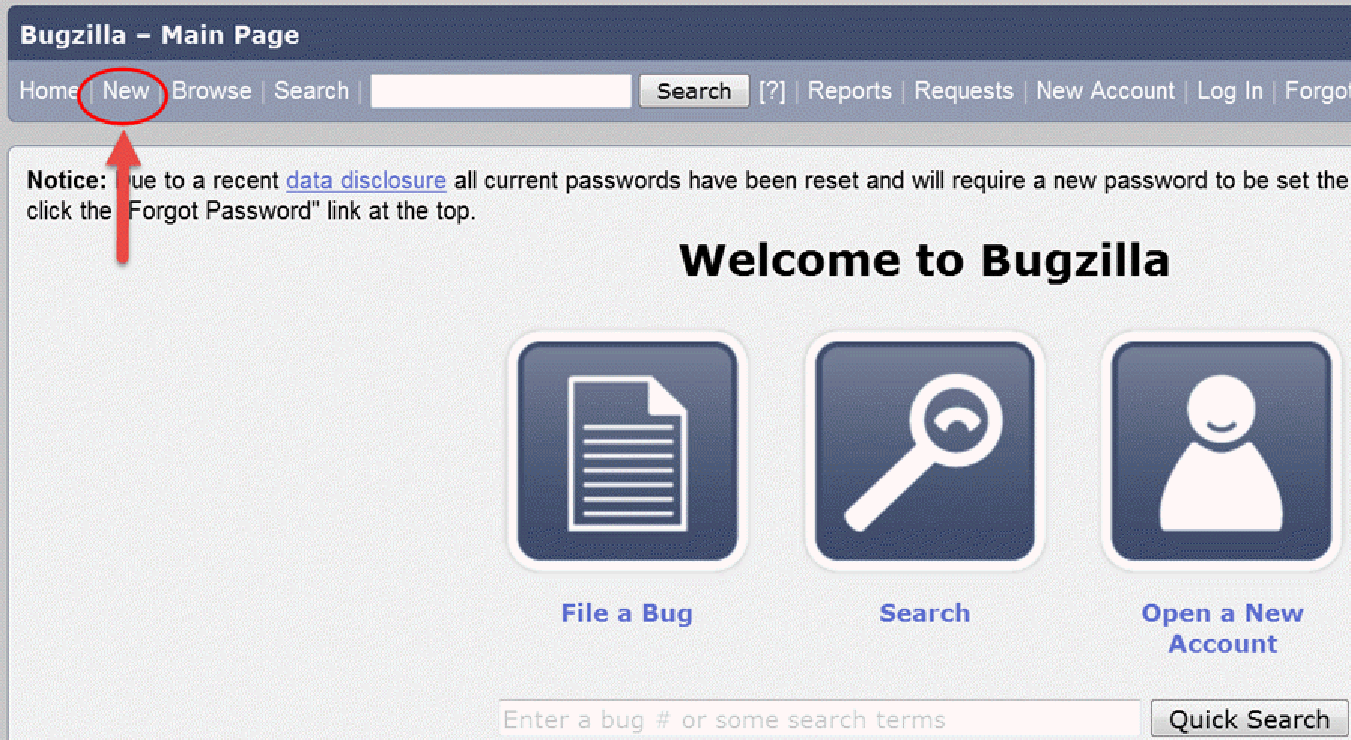
****

Step 3) You are successfully logged into the Bugzilla system.



**Creating a Bug-report in Bugzilla**

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



Step 2) In the next window

Enter Product

Enter Component

Give Component description

Select version,

Select severity

Select Hardware

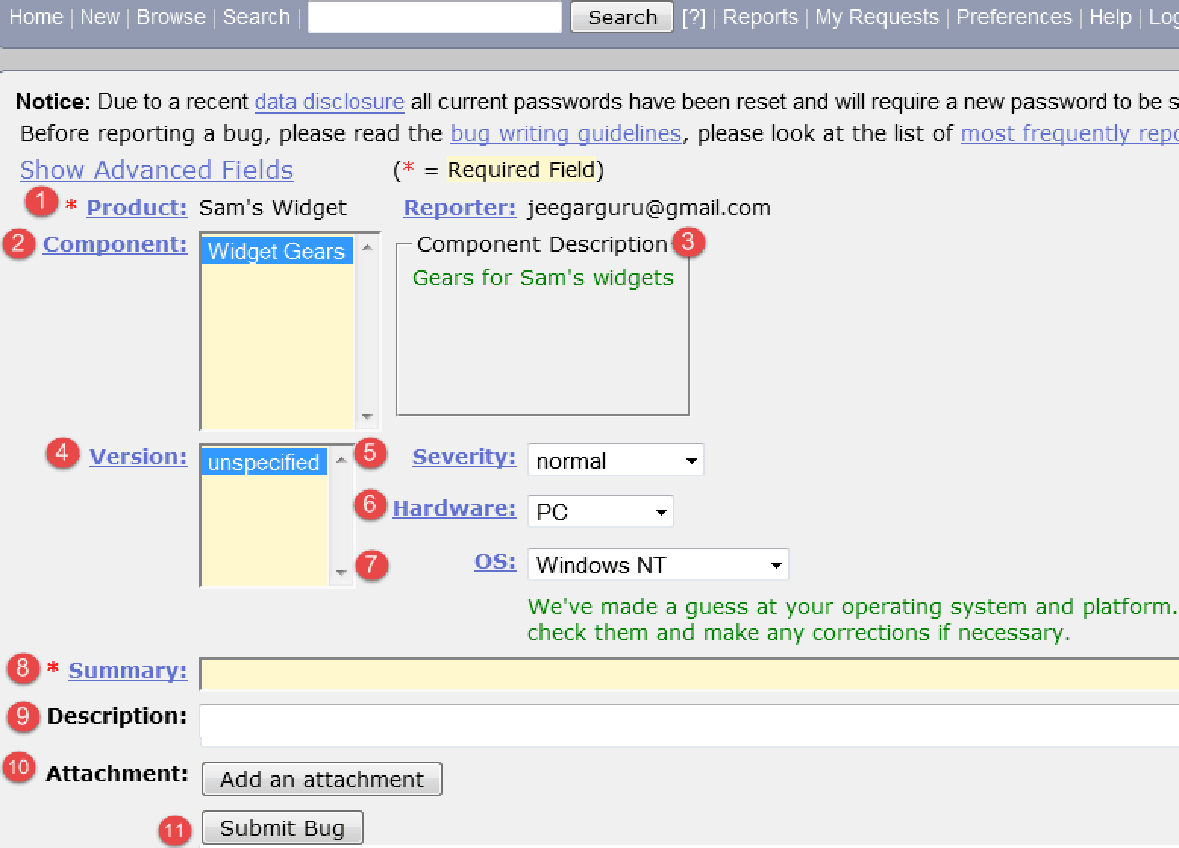
Select OS

Enter Summary

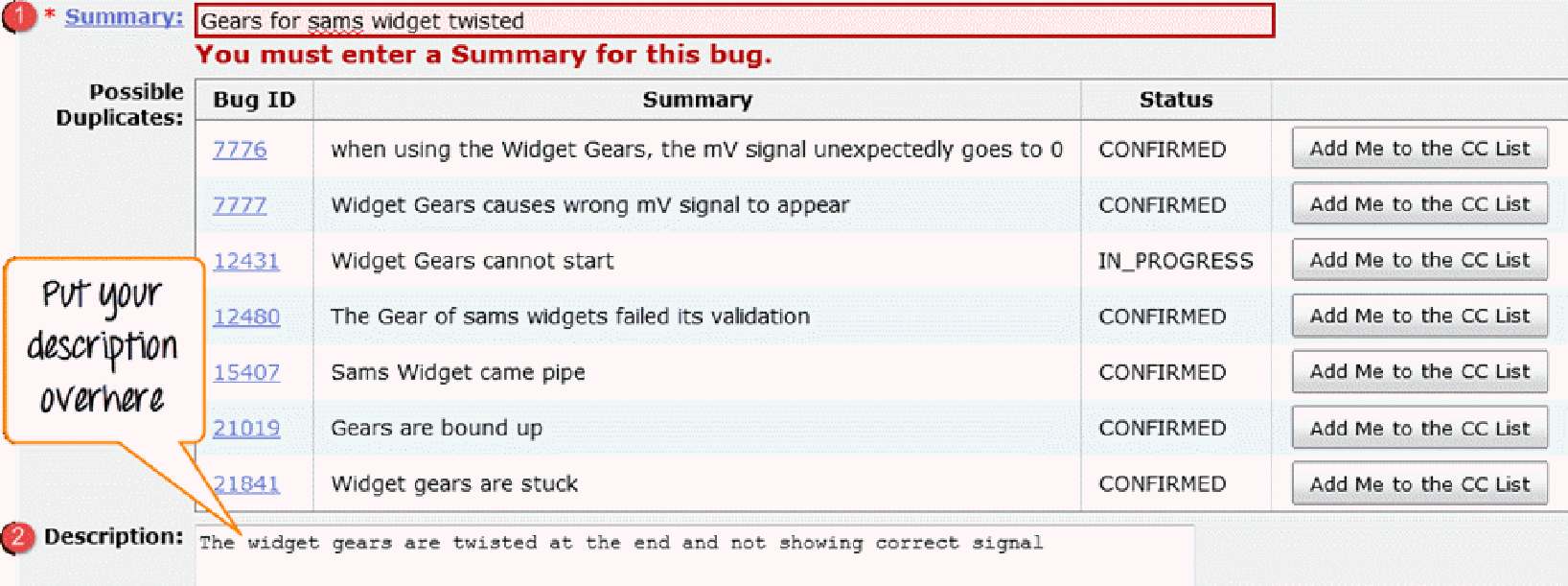
Enter Description

Attach Attachment

Submit

****

If you don’t fill the mandatory fields, you get the below window.

****

Step 4) Bug is created ID# 26320 is assigned to our Bug. You 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 you have created.

Large text box

URL

Whiteboard

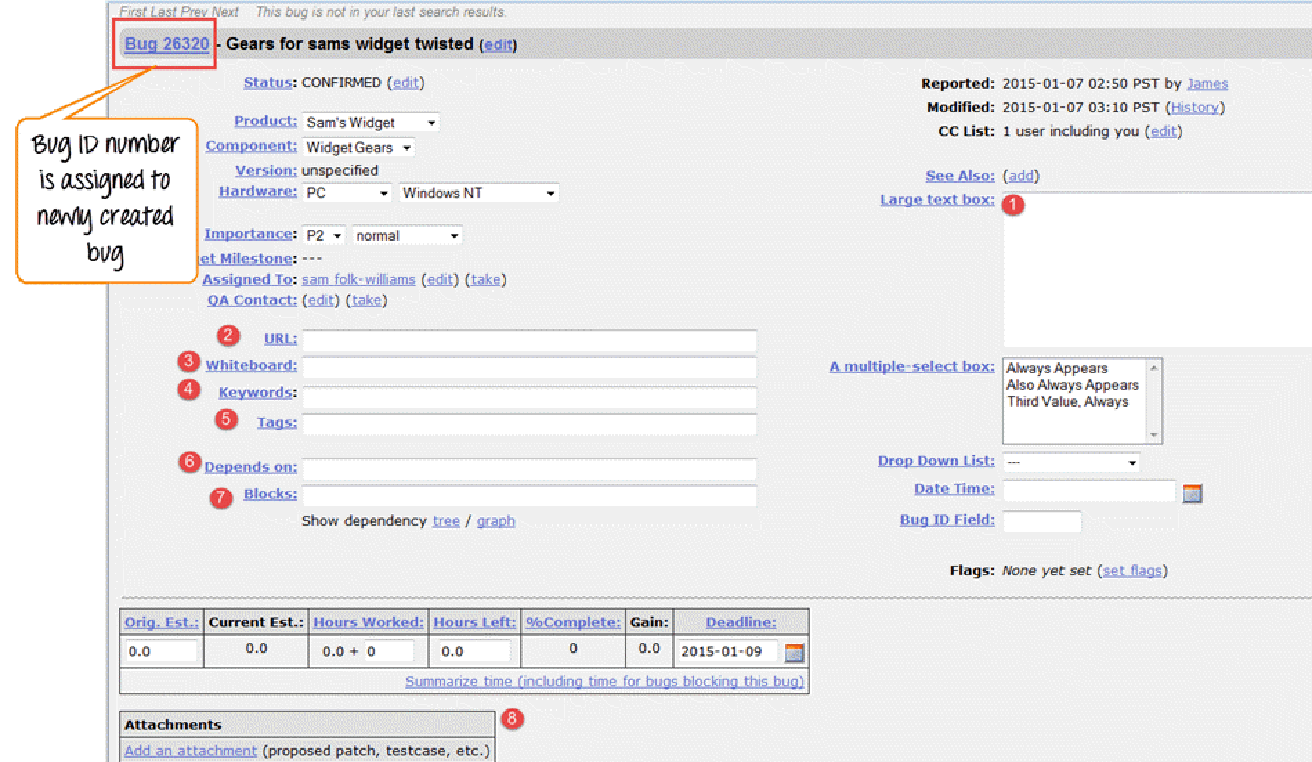
Keywords

Tags

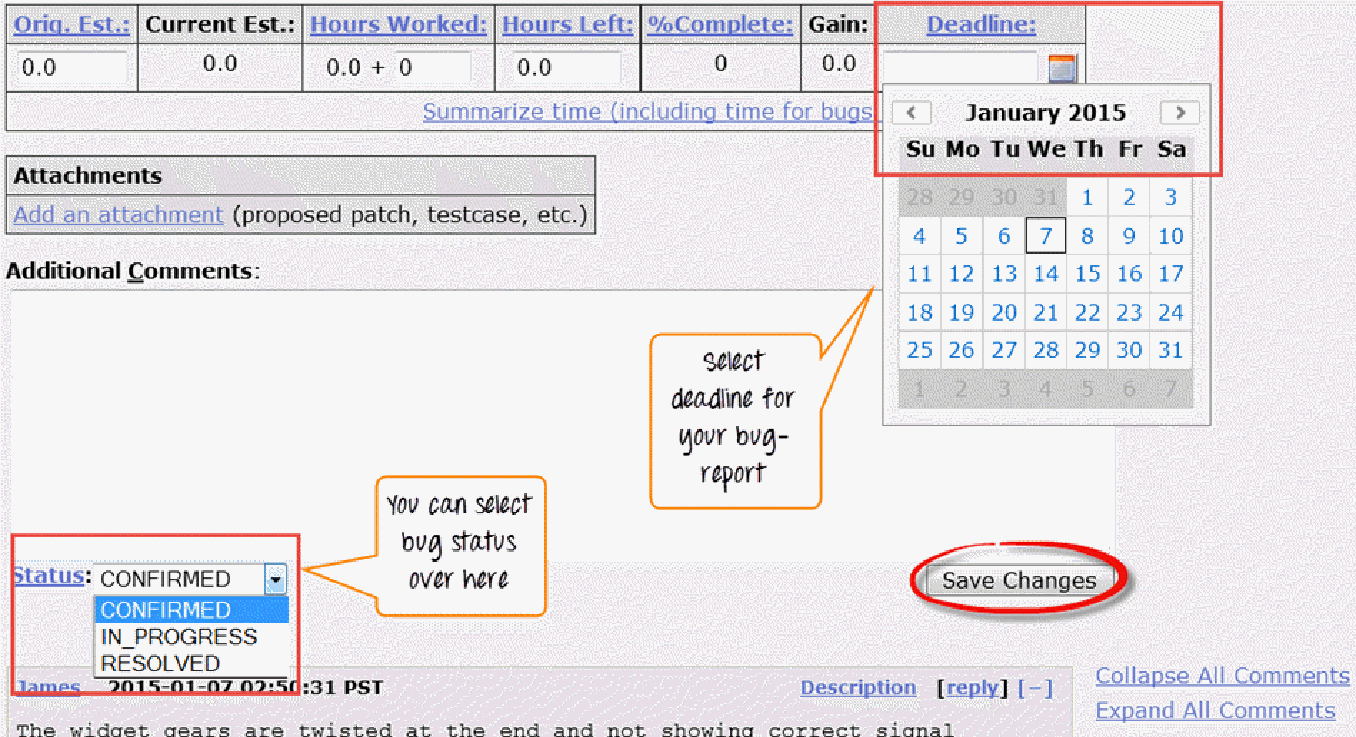
Depends on

Blocks

Attachments

****

Step 5) In the same window if you scroll down further. You can select the deadline date and also the status of the bug. The deadline in Bugzilla usually gives the time limit to resolve the bug in the given time frame.

****

**Create Graphical Reports**

Graphical reports are one way to view the current state of the bug database. You can run reports

either through an HTML table or graphical line/pie/bar-chart-based one. The idea behind the

graphical report in Bugzilla is to define a set of bugs using the standard search interface and then

choose some aspect of that set to plot on the horizontal and vertical axes. You can also get a

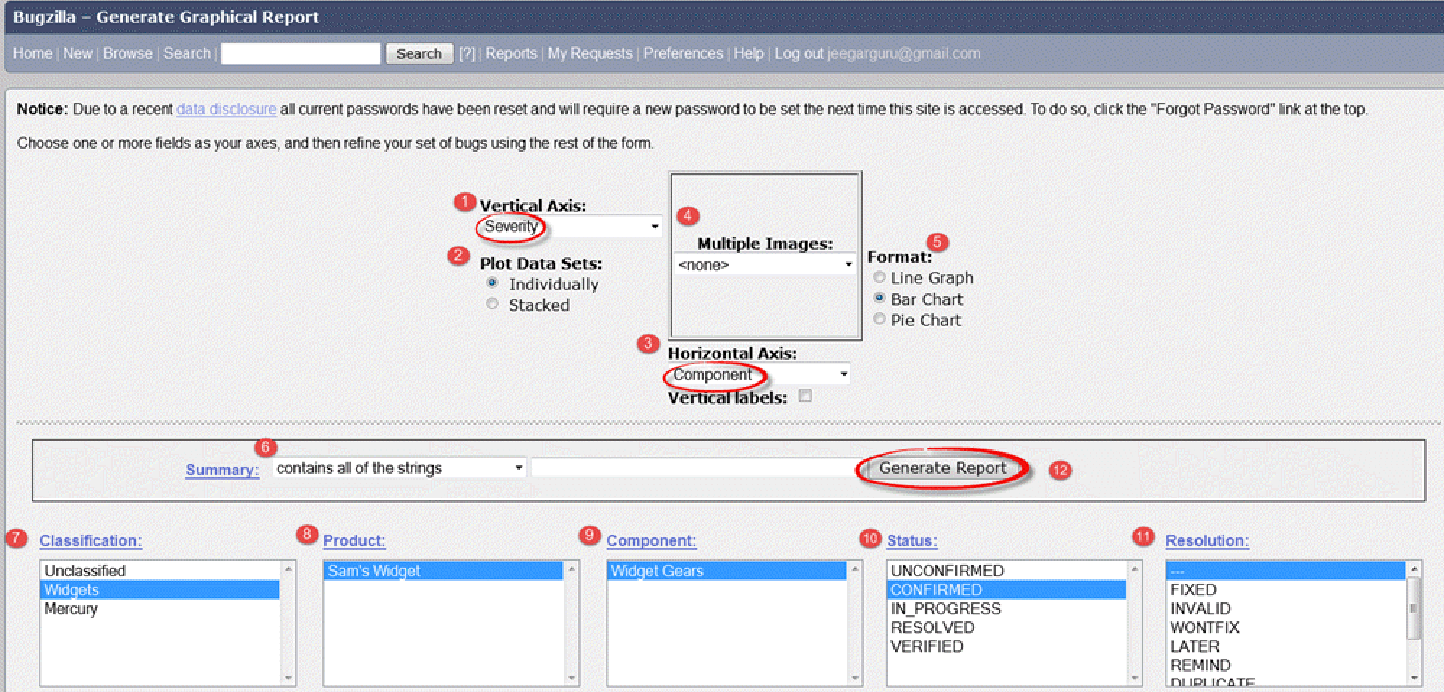
3-dimensional report by choosing the option of “Multiple Pages”.

Reports are helpful in many ways, for instance, if you want to know which component has the

largest number of bad bugs reported against it. In order to represent that in the graph, you can

select severity on X-axis and component on Y-axis and then click on generate a report. It will

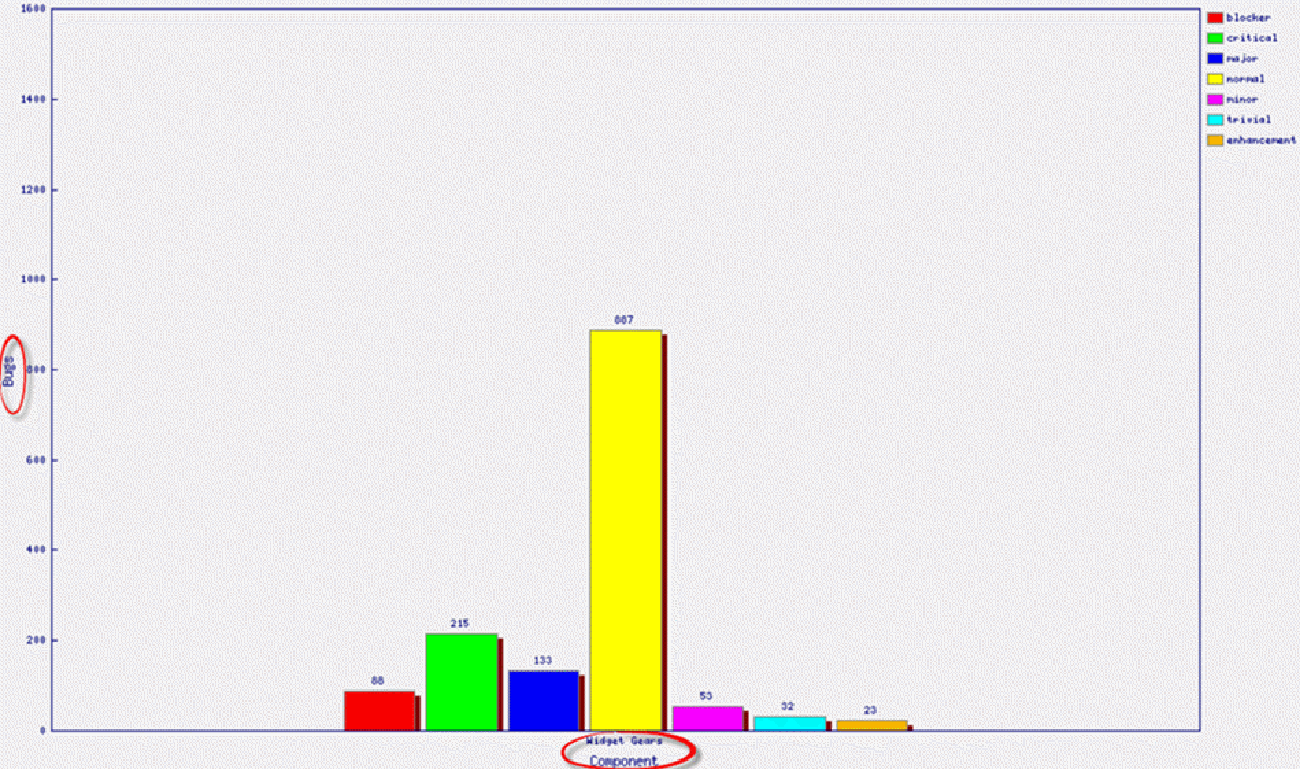
generate a report with crucial information.

****

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 the top with 667 numbers.

****

**Jira**

JIRA is a tool developed by Australian Company Atlassian. It is used for bug tracking, issue tracking, and project management. The name "JIRA" is actually inherited from the Japanese word "Gojira" which means "Godzilla". The basic use of this tool is to track issue and bugs related to your software and Mobile apps. It is also used for project management. The JIRA dashboard consists of many useful functions and features which make handling of issues easy. Some of the key features are listed below. Let's learn JIRA Defect and Project tracking software with this Training Course.

**JIRA Scheme**

Inside JIRA scheme, everything can be configured, and it consists of

* Workflows
* Issue Types
* Custom Fields
* Screens
* Field Configuration
* Notification
* Permissions

**What is JIRA Issue?**

JIRA issue would track bug or issue that underlies the project. Once you have imported a project then you can create issues. Under Issues, you will find other useful features like

* Issue Types
* Workflow's
* Screens
* Fields
* Issue Attributes

**Issue Types**

Issue Type displays all types of items that can be created and tracked via JIRA. JIRA

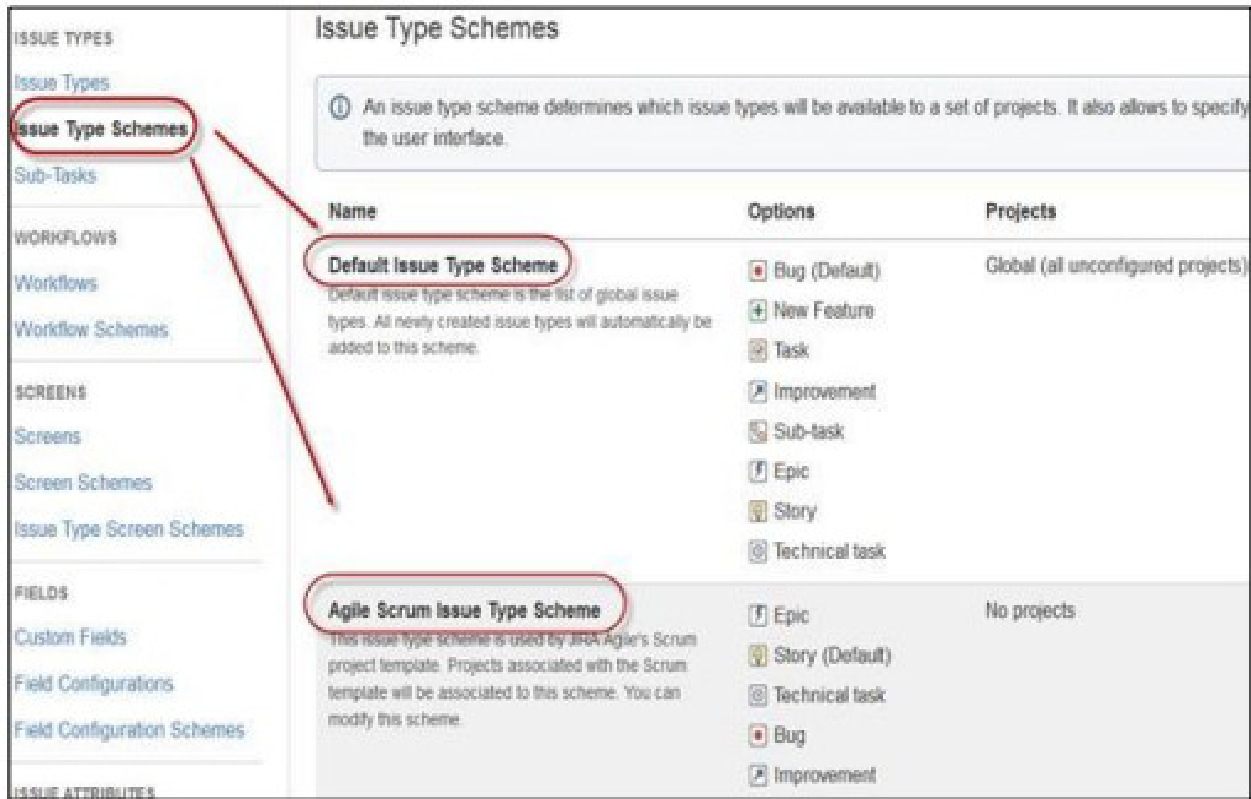
Issues are classified under various forms like new feature, sub-task, bug, etc. as shown

in the screen shot.

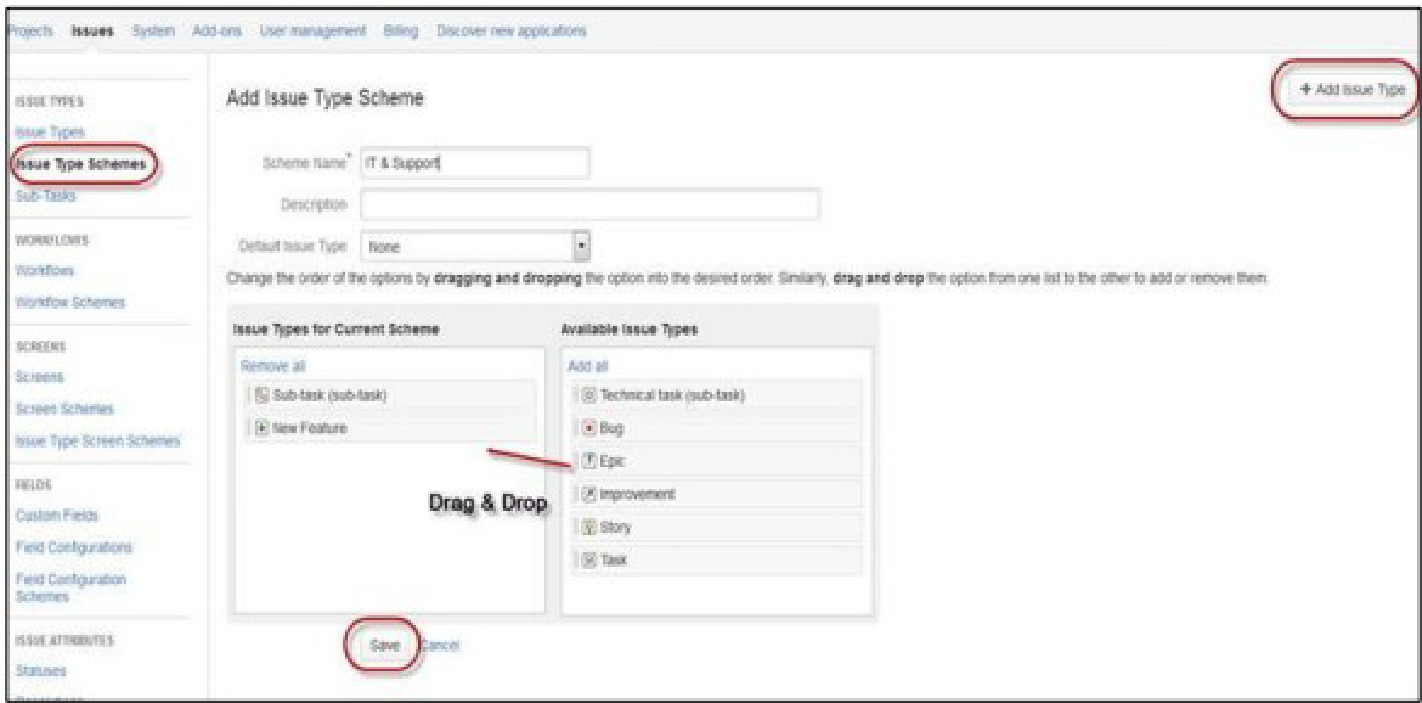


There are two types of Issue types schemes in JIRA:

* **Default Issue Type Scheme:** In default issue type scheme all newly created issues will be added automatically to this scheme.
* **Agile Scrum Issue Type Scheme:** Issues and project associated with Agile Scrum will use this scheme.



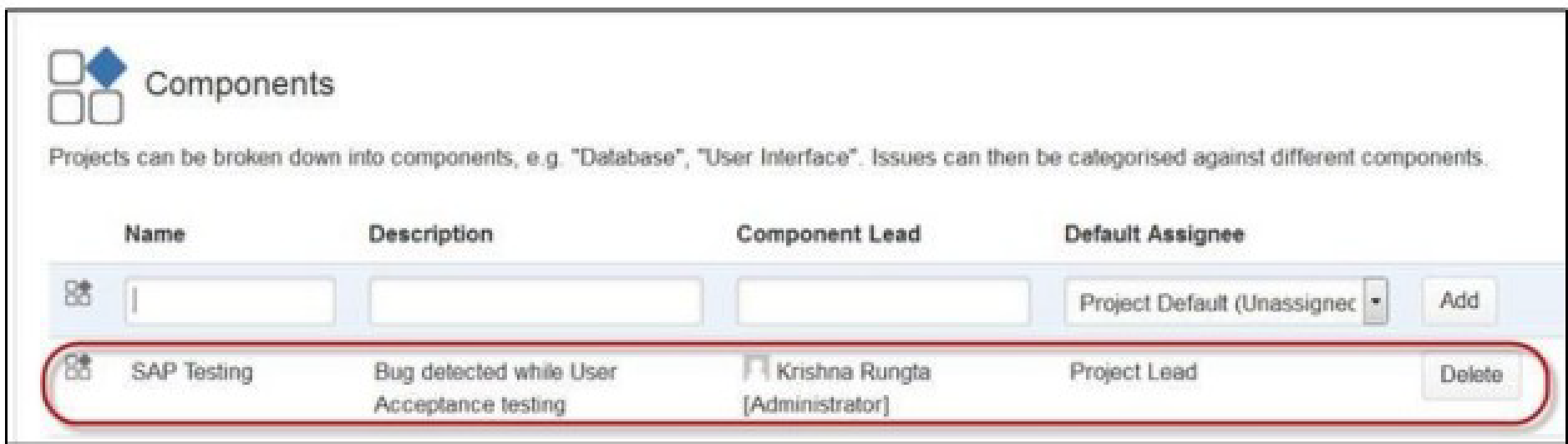
Apart from these two issue type schemes, you can also add schemes manually as per requirement, for example we have created IT & Support scheme, for these we will drag and drop the issue types from the Available Issue type to Issue type for current scheme as shown in the screenshot below.

****

**JIRA Components**

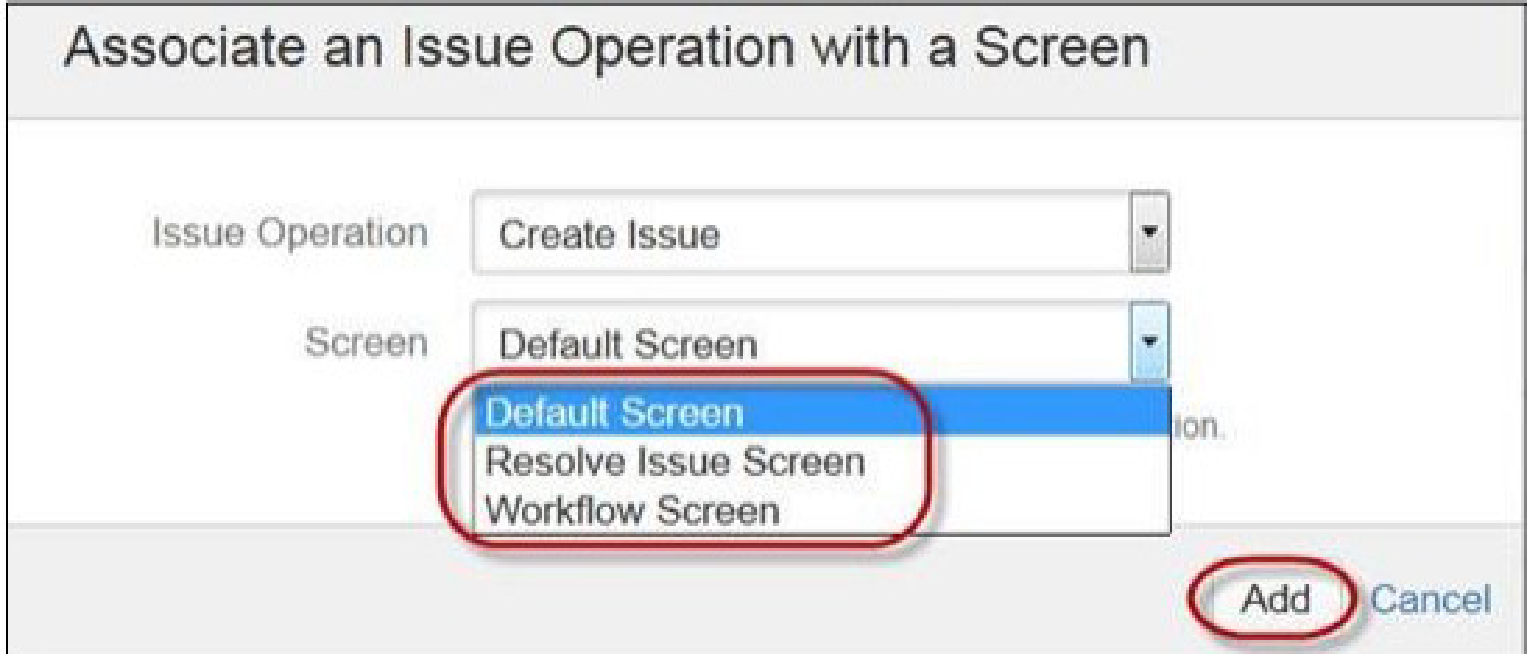
Components are sub-sections of a project; they are used to group issues within a project into smaller parts. Components add some structures to the projects, breaking it up into features, teams, modules, subprojects and more. Using components, you can generate reports, collect statistics, and display it on dashboards and so on.

To add new components, as shown in the below screen you can add name, description, component lead and default assignee.

****

**JIRA screen**

When an issue is created in JIRA, it will be arranged and represented into different fields, this display of fields in JIRA is known as a screen. This field can be transitioned and edited through workflow. For each issue, you can assign the screen type as shown in the screen-shot. To add or associate an issue operation with a screen you have to go in the main menu and click on Issues then click on Screen Schemes and then click on "Associate an issue operation with a screen" and add the screen according to the requirement.

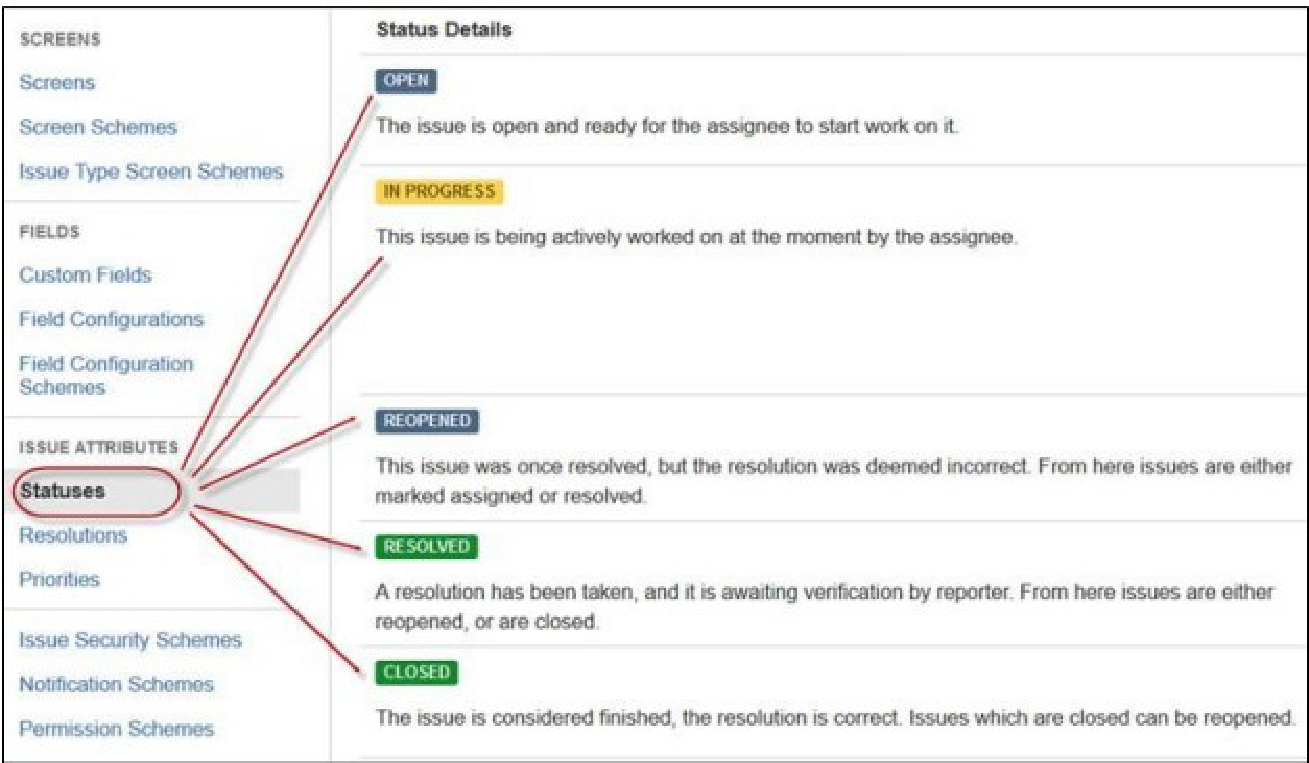


**Issue Attributes**

Issue Attributes encompasses

* Statuses
* Resolutions
* Priorities

Statuses: Different statuses are used to indicate the progress of a project like To do, InProgress, Open, Closed, ReOpened, and Resolved. Likewise, you have resolutions and priorities, in resolution it again tells about the progress of issue like Fixed, Won't fix, Duplicate, Incomplete, Cannot reproduce, Done also you can set the priorities of the issue whether an issue is critical, major, minor, blocker and Trivial.



**Issue Security Schemes**

This function in JIRA allows you to control who can view the issues. It consists of a number of security levels which can have users or groups assigned to them. You can specify the level of security for the issues while creating or editing an issue.

Similarly, there is a Default Permission Scheme any new project that is created will be assigned to this scheme. Permission Schemes allow you to create a set of permissions and apply this set of permission to any project.

**Viva Questions:**

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

**Ans.** Organizations involved in multiple software development projects would find Bugzilla very helpful in their efforts, as it provides them with a convenient centralized location for tracking ongoing initiatives and helps cut down on duplicate and unnecessary work. All members of the DevOps team, no matter what their involvement is during development, will find the software simple to use. To help maintain a hierarchy, the Bugzilla tool offers two access modes: user and administrator. The administrator role provides users with greater access to security, workflow monitoring and grouping features.

**Q2. Why is JIRA used?**

**Ans.** Jira Software is part of a family of products designed to help teams of all types manage work.

Originally, Jira was designed as a bug and issue tracker. But today, Jira has evolved into a powerful work management tool for all kinds of use cases, from requirements and test case management to agile software development.

**Q3. What is the deployment environment for Bugzilla?**

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

**Q4. What is Bugzilla?**

**Ans.** Bugzilla is a Web-based Defect Management Tool that allows Testing and Development teams to

post and track defects and it is an Open source tool.

**Q5. What are Bugzilla features?**

**Ans.** The Bugzilla features are as follows:

* Integrated, product-based granular security schema
* Inter-bug dependencies and dependency graphing
* Advanced reporting capabilities
* A robust, stable RDBMS back-end
* Extensive configurability
* A very well-understood and well-thought-out natural bug resolution protocol
* Email, XML, console, and HTTP APIs

**Experiment – 10**

**Aim:** Study of any open source testing tool (OSTA).

**Theory:**

**Introduction**

OpenSTA supplies versatile Test development software that enables you to create and run Tests tailor-made for the environment you are assessing. The contents and structure of a Test will depend on the type of test you are conducting, the nature of the system you are targeting and the aims of your performance test.

OpenSTA supports the creation of HTTP/S load Tests that include Scripts which supply the load element and may also include Collectors which are used to record additional performance data. You can also use OpenSTA to develop Collector-only used Tests for monitoring production environments. It is possible to use the same Test in both these scenarios.

Running a Test with both Script and Collector Task Groups enabled allows you to test and record the performance of a WAE during development. After the WAE goes live, Script-based Task Groups can be disabled and the Test re-run within a production environment. This enables you to generate useful comparative performance results and to continuously monitor the target WAE.

The example in this tutorial works through the creation of an HTTP/S load Test which includes Script-based and Collector-based Task Groups.

**OpenSTA Architecture**

OpenSTA supplies a distributed software testing architecture based on CORBA which enables you to create then run Tests across a network. The OpenSTA Name Server configuration utility is the component that allows you to control your Test environment.

After installing OpenSTA you will notice that the OpenSTA Name Server is running indicated by , in the Windows Task Bar. This component must be running before you can run a Test.

If no icon appears click **Start > Programs > OpenSTA > OpenSTA NameServer.**

If the OpenSTA Name Server stops the Name Server Configuration utility icon

appears , in the Task Bar. You can start it by right-clicking and selecting Start Name

Server from the menu.

**Commander**

Commander is the Graphical User Interface that runs within the OpenSTA Architecture and functions as the front end for all Test development activity. It is the program you need to run in order to use OpenSTA.

Launch Commander

Click **Start > Programs > OpenSTA > OpenSTA Commander.**

**The Commander Interface**

Commander combines an intuitive user interface with comprehensive functionality to give you control over the Test development process, enabling you to successfully create and run HTTP/S performance Tests.

Use the menu options or work from the Repository Window to initiate the creation of Collectors and Tests. Right-click on the predefined folders in the Repository Window and choose from the functions available.

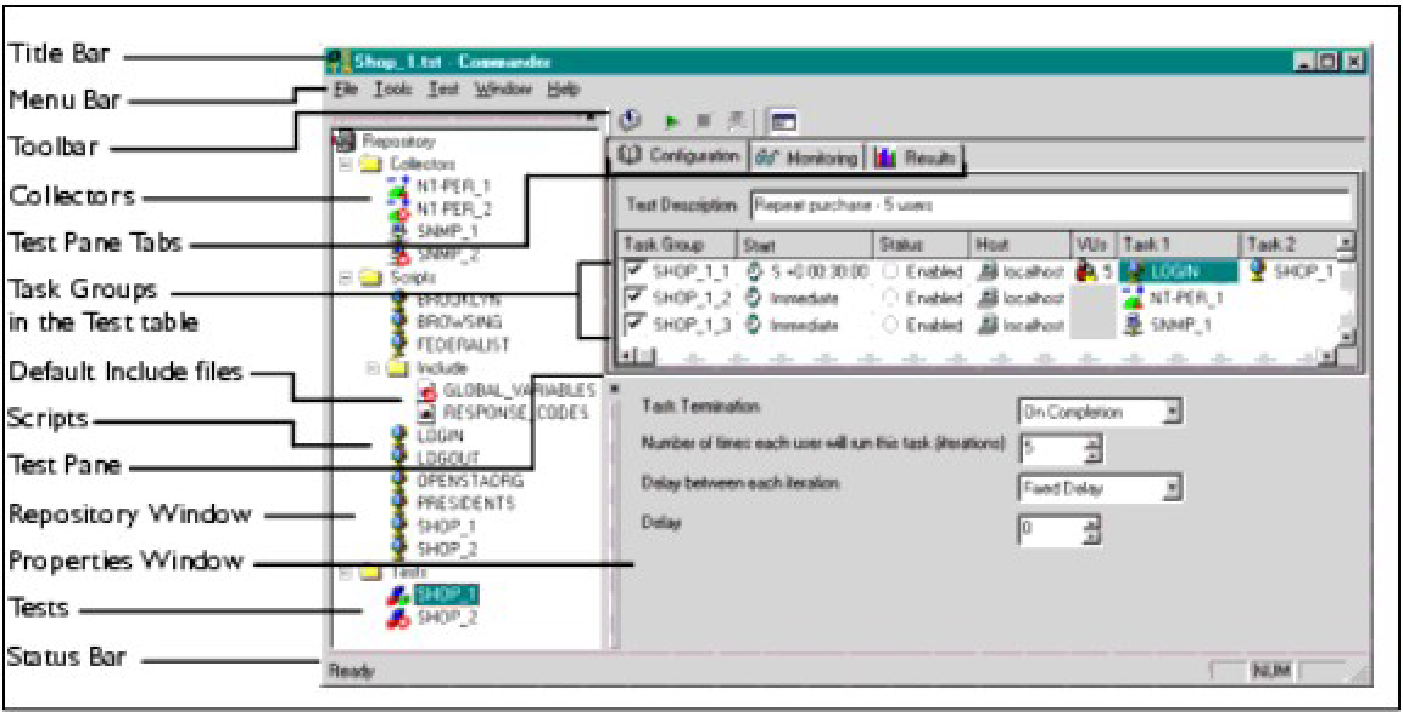
Work within the Main Window of Commander to create Collectors and Tests. The Main Window houses the Repository Window and supplies the workspace for Test creation using the Test Pane, and Collector creation using the Collector Pane. Use Script Modeler to create the Scripts you need.

After you have created or edited a Test or Collector in the Main Window it is automatically saved when you switch to another procedure or exit from Commander.

**Commander Interface Features**

The Commander interface is divided up into three primary areas:

* Commander Toolbars and Function Bars.
* The Repository Window.
* The Commander Main Window.



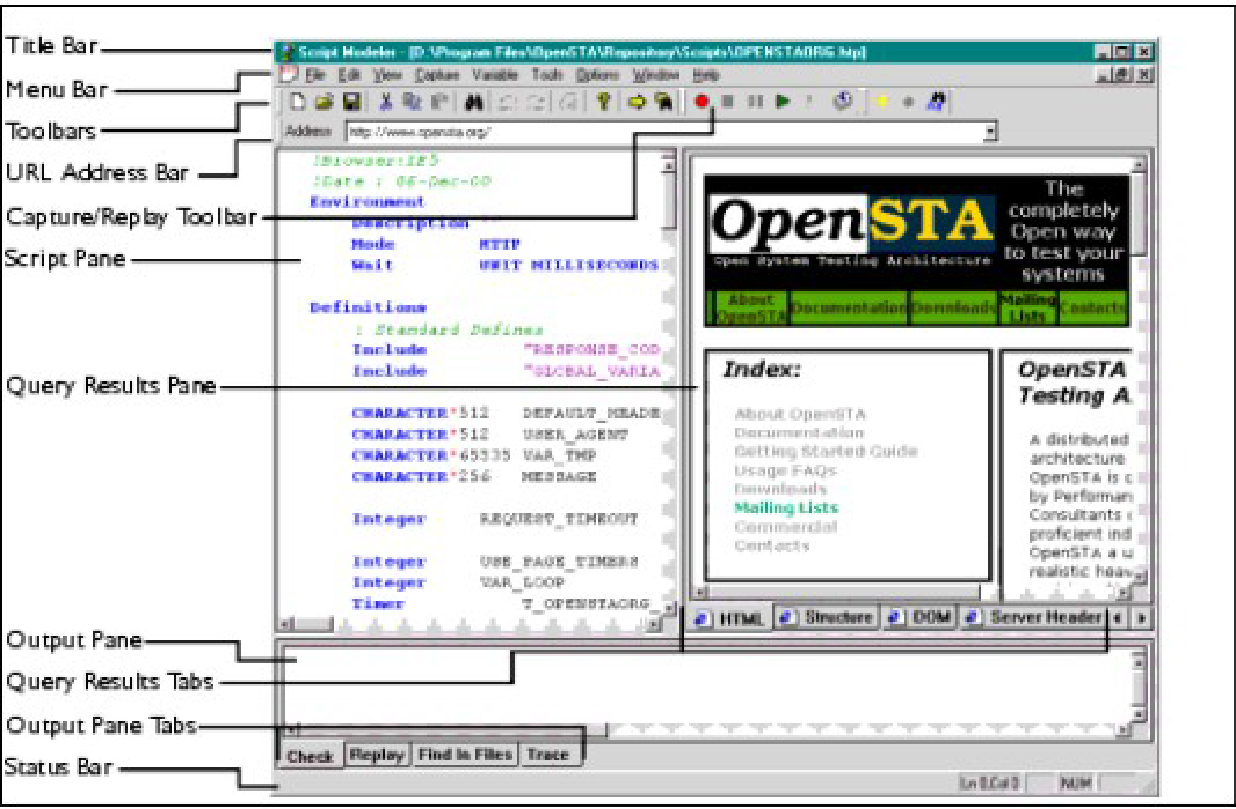
**Script Modeler**

Script Modeler supplies versatile Script creation and modeling functionality. Use the menu bar and right-click menu options to create and model Scripts.

After you create a Script or when you open one, it is displayed in the Script Pane on the left-hand side of the main window. It is represented using SCL code which enables you to model it using the menu options or directly by keying in the SCL commands you need.

The Query Results Pane is used to display Web site responses. HTML information is recorded during the same Web session as the corresponding Script and is directly related to it, which enables additional modeling options.

The Output Pane. Displays the results of Script compilation. Scripts need to be compiled after modeling to check the validity.



**The Test Pane**

Use the Test Pane to create and edit a Test, then apply the Task Group settings you require to control how they behave during a Test-run. Run and monitor the Test-run then display your results for analysis.

The Test Pane is displayed in the Main Window when you open a Test by double clicking a new Test , or an existing Test , in the Repository Window.

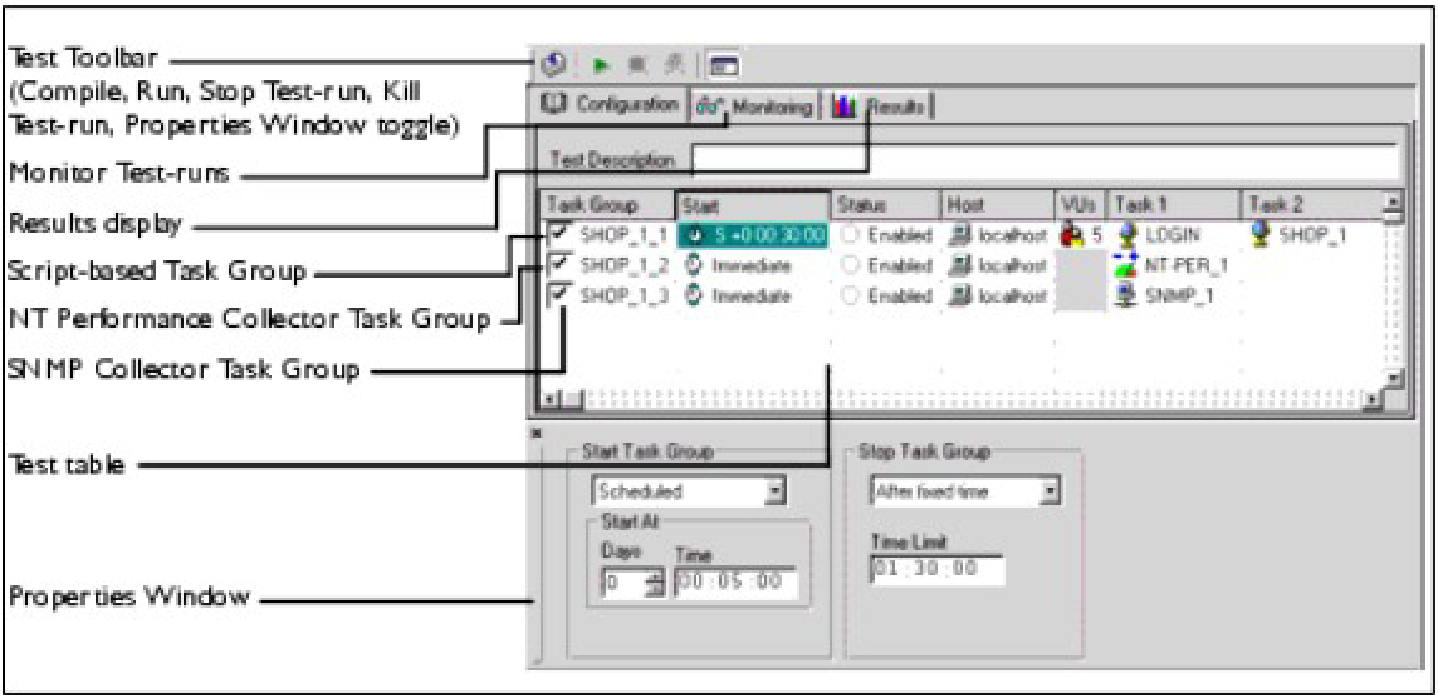
The Test Pane comprises three sections represented by the following tabs:

* **Configuration:** This is the default view when you open a Test and the workspace used to develop a Test. Use it in combination with the Repository Window to select and add Scripts and Collectors. It displays the Test table which has categorized column headings that indicate where Script and Collector Tasks can be placed and the Task Group settings that apply to the contents of the Test.

Select a Task Group cell to view and edit the associated settings using the Properties Window displayed below the Test table.

* **Monitoring:** Use this tab to monitor the progress of a Test-run. Select the display options you want from the Monitoring Window, including a Summary and data for individual Task Groups.
* **Results:** Use this tab to view the results collected during Test-runs in graph and table format. Use the Results Window to select the display options available which are dependent on the type of Test you are running.

**Test pane Features**



**The Test Development Process**

The Test development process typically includes the following procedures:

* Create a Test
* Add a Script to a Test
* Add Collectors to a Test
* Define Task Group settings, these include:
* Edit the Task Group Schedule Settings
* Select the Host used to Run a Task Group
* Specify the Number of Virtual Users to run a Script-based Task Group
* Edit the Number of Script Iterations and the Delay Between Iterations
* Save and Close a Test

**Viva Questions:**

**Q1. Why choose Open Source Performance Test tool?**

**Ans.** Open source load testing tools are often maintained by some of the big developers and are free to use with their source code available to customize as required.

Advantages of using open source load testing tools

* No initial investment is needed, as it is free to use.
* The source code is available to extend the functionality based on internal use cases.
* Managed by top developers and open source communities.

**Q2. Explain what is performance testing?**

**Ans.** Performance testing can be used to analyze various success factors such as response times and potential errors. With these performance results in hand, you can confidently identify bottlenecks, bugs, and mistakes – and decide how to optimize your application to eliminate the problems.

**Q3. Mention different types of performance testing?**

1. Stress Testing.
2. Spike Testing.
3. Load Testing.
4. Endurance Testing.
5. Volume Testing.
6. Scalability Testing.

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

**Ans.**  Some common problems are:

1. Poorly written code.
2. Non-optimized databases.
3. DNS, firewall, and network connectivity.
4. Troublesome third-party services.

**Q5. List out some of the performance testing tool?**

**Ans.** Some performance testing tools are:

1. WebLOAD.
2. LoadNinja.
3. HeadSpin.
4. ReadyAPI Performance.
5. LoadView.
6. Keysight's Eggplant.
7. Apache JMeter.
8. LoadRunner.

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

**Ans.** Some common performance bottlenecks are:

1. CPU Utilization.
2. Memory Utilization.
3. Network Utilization.
4. Software Limitation.
5. Disk Usage.

**Experiment – 11**

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

**Theory:**

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.

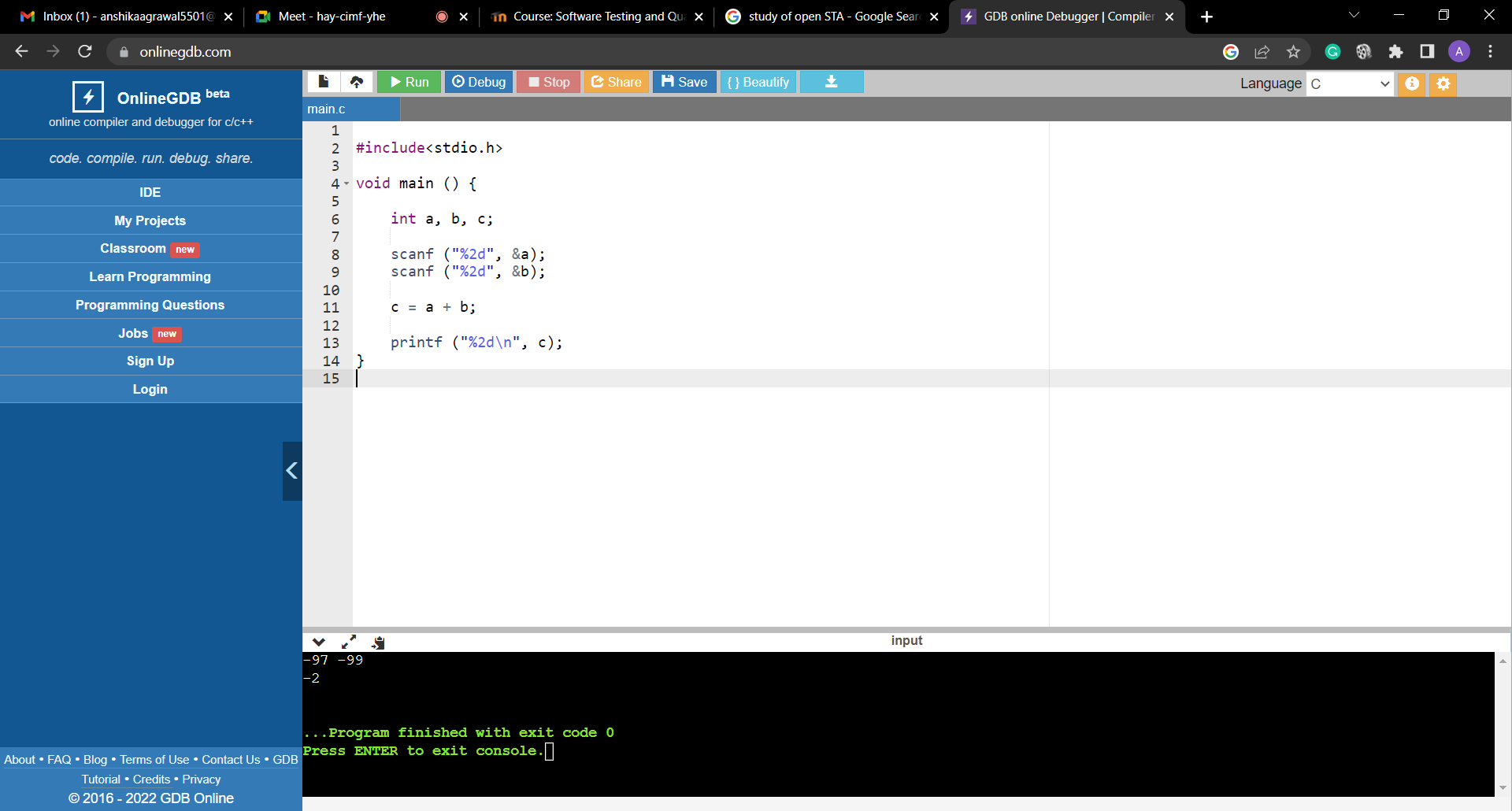
**Some Advantages –**

1. Ad-hoc testing gives freedom to the tester to apply their own new ways of testing the application which helps them to find out more number of defects compared to the formal testing process.
2. This type of testing can be done at anytime anywhere in the Software Development Life cycle (SDLC) without following any formal process.
3. This type of testing is not only limited to the testing team but this can be done by the developer while developing their module which helps them to code in a better way.

**Some Disadvantages –**

1. Since ad-hoc testing is done without any planning and in unstructured way so recreation of bugs sometime becomes a big trouble.
2. The test scenarios executed during the ad-hoc testing are not documented so the tester has to keep all the scenarios in their mind which he/she might not be able to recollect in future.
3. Ad-hoc testing is very much dependent on the skilled tester who has thorough knowledge of the product it cannot be done by any new joiner of the team.

**Code:**



**Results:**

|  |  |  |  |
| --- | --- | --- | --- |
| **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 |
| 4 | -38, 99 | 61 | 5 |
| 5 | 20,30 | 50 | 50 |
| 6 | 53, 45 | 98 | 98 |
| 7 | 10,77 | 87 | 87 |
| 8 | 1,1 | 2 | 2 |
| 9 | -34,90 | 56 | 1 |
| 10 | 11,12 | 23 | 23 |
| 11 | -45,78 | 23 | 1 |
| 12 | 72,34 | 106 | 106 |
| 13 | -99,55 | -44 | 0 |

**Viva Questions:**

**Q1.** **What exactly ad-hoc testing is?**

**Ans.** Ad hoc Testing is an informal or unstructured software testing type that aims to break the testing process in order to find possible defects or errors at an early possible stage. Ad hoc testing is done randomly and it is usually an unplanned activity which does not follow any documentation and test design techniques to create test cases.

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

**Ans.** Ad hoc testing can be performed when there is limited time to do elaborative testing. Usually Adhoc testing is performed after the formal test execution. And if time permits, ad hoc testing can be done on the system. Ad hoc testing will be effective only if the tester is knowledgeable of the System Under Test.

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

**Ans.** Following are some drawbacks:

1. Since ad-hoc testing is done without any planning and in unstructured way so recreation of bugs sometime becomes a big trouble.
2. The test scenarios executed during the ad-hoc testing are not documented so the tester has to keep all the scenarios in their mind which he/she might not be able to recollect in future.
3. Ad-hoc testing is very much dependent on the skilled tester who has thorough knowledge of the product it cannot be done by any new joiner of the team.

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

**Ans.**  Following are some advantages:

1. Ad-hoc testing gives freedom to the tester to apply their own new ways of testing the application which helps them to find out more number of defects compared to the formal testing process.
2. This type of testing can be done at anytime anywhere in the Software Development Life cycle (SDLC) without following any formal process.
3. This type of testing is not only limited to the testing team but this can be done by the developer while developing their module which helps them to code in a better way.

**Experiment – 12**

**Aim:** 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 (“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

**Theory:**

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.

* 1. 4. Test cases that exercise internal data structures to ensure their validity.

Following are the white box testing techniques:

1. **Basis Path testing**

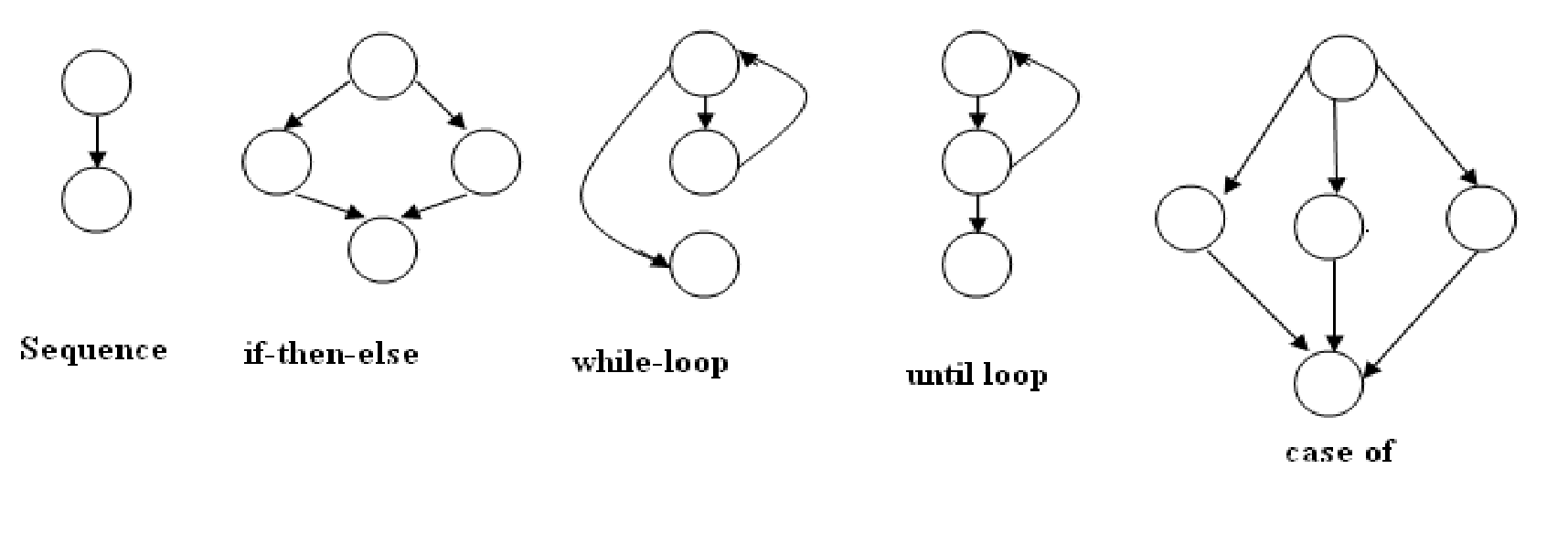
With the help of basis path testing, test case designer can derive a logical complexity measure of a procedural design and can use this measure to find the basis set of execution paths. Test cases that are derived to exercise the basis set ensures that every statement in the program will execute at least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

1. **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



Cyclomatic complexitycan 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.

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

V(G) = P+1

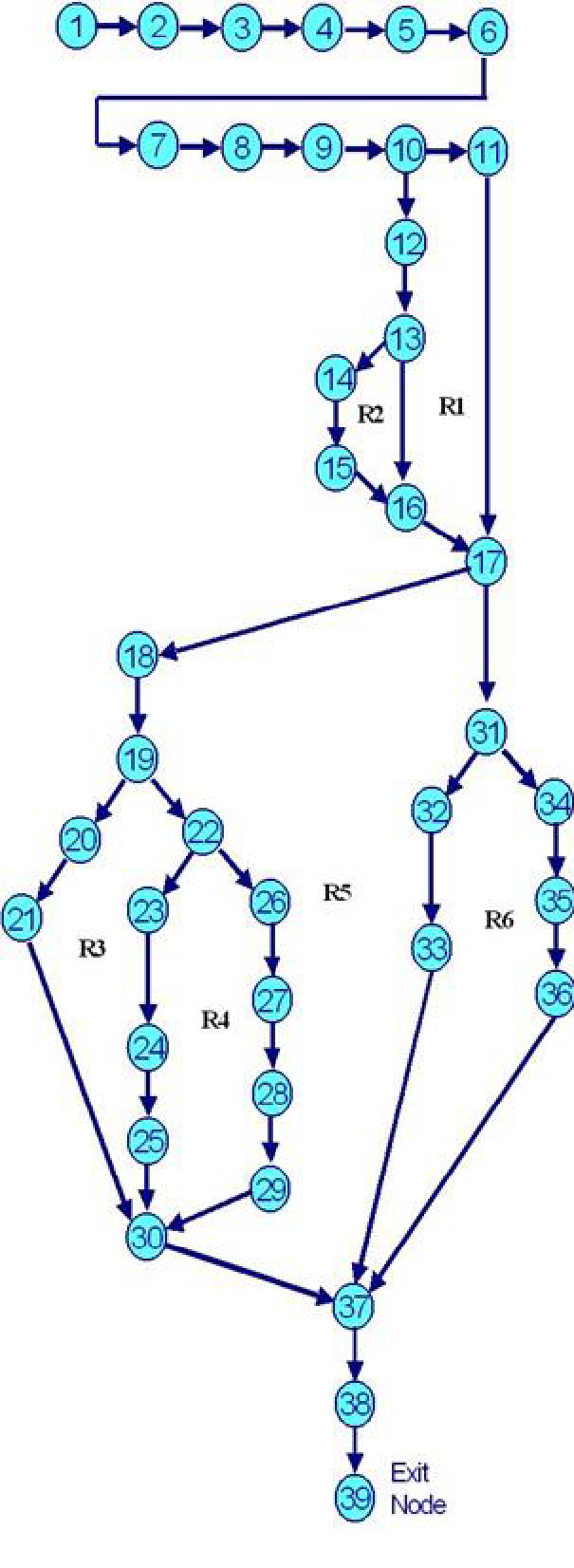
Where P is the number of predicate node in the flow graph

1. **Independent path**

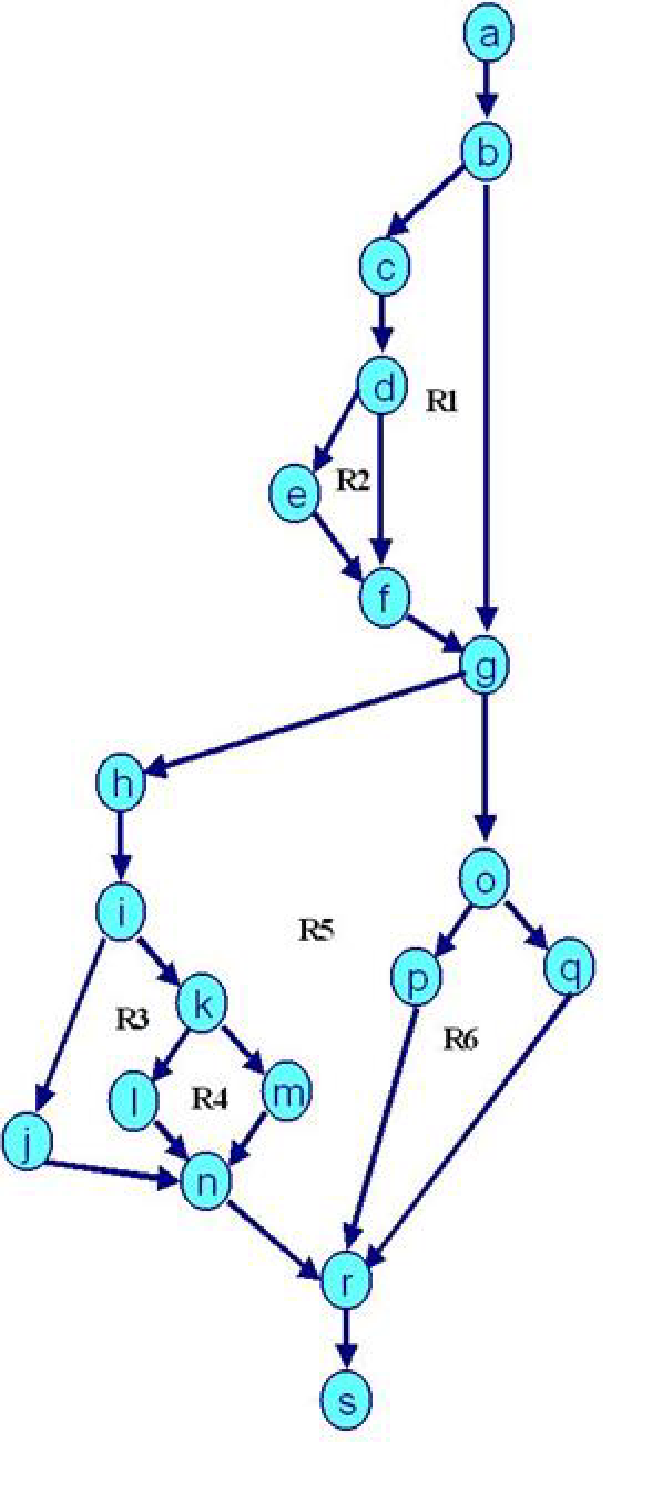
Number of independent path = V(G) which makes a basis set. We must execute these paths at least once in order to test the program thoroughly.

**Result:**

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

****

1. **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 |

1. **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**.**

1. **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.**

**Ans.**

|  |  |
| --- | --- |
| **Black box testing** | **White box testing** |
| It is a way of software testing in which the internal structure or the program or the code is hidden and nothing is known about it. | It is a way of testing the software in which the tester has knowledge about the internal structure or the code or the program of the software. |
| Implementation of code is not required. | Implementation of code is required. |
| It is mostly done by software testers. | It is mostly done by software developers. |
| It is a functional test of software. | It is a structural test of software. |

**Q2. Define cyclomatic complexity.**

**Ans.** Cyclomatic complexity (CYC) is a software metric used to determine the complexity of a program. It is a count of the number of decisions in the source code. The higher the count, the more complex the code.

**Q3. What is DD Path Testing.**

**Ans.** When we have a flow graph, we can easily draw another graph that is known as decision-to-decision or (DD) path graph, wherein we lay our main focus on the decision nodes only. The nodes of the flow graph are combined into a single node it they are in sequence.

**Q4. Why we do White Box Testing?**

**Ans.** It is essential because it helps to test the following:

* It is executed at different levels such as system, integration, and unit level of software development.
* One primary goal of it is to verify the working of an application.
* It involves the identification of the operational flow of an application.

**Experiment – 13**

**Aim:** Study of open source automated testing suite for web applications (Selenium tool).

**Theory:**

Selenium is a robust set of tools that support rapid development of test automation for web based applications.

**Selenium Components**

* Selenium is composed of 4 major tools:
  + IDE
  + RC
  + Webdriver
  + Grid
* Selenium RC provides on API and library for each of its supported languages HTML, JAVA, C#, Pearl, Ruby

**Selenium grids**

It allows the selenium RC solution to seek for largest test cases.suites that must run in multiple environments with selenium RC are running on various operating system and browser configuration, each of those when launching register with a hub

**Flexibility and extensibility**

Selenium is highly flexible. Selenium IDE allows for addition of user defined user extensions for creating additional commands customized to the user needs.

**Test Suite**

A test suite is a collection of test. Often one will run all the tests in a test suite as one continuous batch job.

When using Selenium IDE , test suites also can be defined using a simple HTML file. Test suits can also be maintained when using Selenium RC

Few typical Selenium commands are :

* Open - opens a page using URL
* Click - Performs a click
* Verify title - Verifies an expected page title
* Verify test present - Verifies expected test is somewhere on page
* Verify table - Verifies a table’’s contents
* Wait for page to load- pauses execution until one expected new page
* Wait for element present - pauses execution until an expected output / UI designed by its HTML is present on the page.

**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.

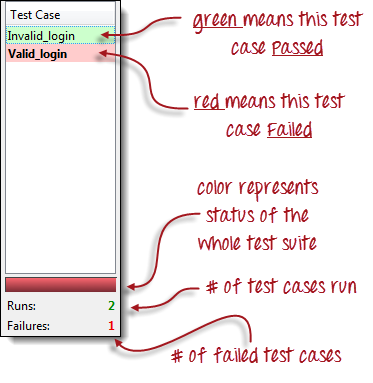
**Features of Selenium IDE**

* 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.

o Green color means "Passed."

o Red color means "Failed."

* 1. • 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?**

**Ans.** Selenium IDE is an open source test automation tool that can record and playback your actions on the web. By using it, you can automate tests for web applications.

**Q2. What is the Test step Syntax?**

**Ans.** Test Step in Selenium IDE Test Case:

Syntax:

Command + Target + Value

Command – forms the functionality of the step

Target – Identifies the Element on the Web page.

Value – Input data (\* This is applicable only for few commands)

**Q3. What are the Advantages of Selenium IDE?**

**Ans.** Benefits of Using Selenium IDE:

1. Provides you the capability of automatically recording your test cases based upon the interactions with the browser
2. Gives developers greater flexibility in executing the test cases. Either the test developer can run the entire test suite consisting of multiple test cases or execute a single test case
3. Allows the test developers to set breakpoints for the purpose of debugging particular test cases.

**Q4. How to install Selenium IDE?**

Step 1: Launch Eclipse.

Step 2: Create Workspace in Eclipse.

Step 3: Create New Java Project in the BrowserStack Workspace.

Step 4: Create Package and Class under the Java Project.

Step 5: Add Selenium JARs to the Java Project in Eclipse.

**Q5. How to create Test cases using Selenium IDE?**

**Ans.** To create a test suite in Selenium IDE:

1. Open Selenium IDE.
2. Go to File>New Test Suite.
3. To add test cases: Go to File>Add test case.
4. Navigate to the location of your test case.
5. Click on Add.
6. Repeat steps 3-5 for more test cases.

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

1. Identify the Selenium tests that you want to run in Test Complete.
2. Create a new Project.
3. Add a Selenium collection to your project.
4. Select which Selenium Framework you are using.
5. Configure your project.
6. Run your test.