

**SRM IST, NCR CAMPUS, MODINAGAR**

# DEPARTMENT OF COMPUTER APPLICATIONS

**PRACTICAL FILE**

**SOFTWARE TESTING.**

**(PCA20D06J)**

# 1st Year, II Semester

**Session: 2022-23**

**Submitted To: Submitted By: Mrs. Beena Sachan. Priy Ranjan**

**(Assistant Professor) REG NO: RA2232241030051**

**SRM INSTITUTE OF SCIENCE &**

**TECHNOLOGY NCR CAMPUS, MODINAGAR**

**DEPARTMENT OF COMPUTER APPLICATIONS**

**RA2232241030051**

**Register No:**

**BONAFIDE CERTIFICATE**

Certified to be the bonafide record of the work done by **Priy Ranjan** of M.C.A, the First year, 2nd Semester for the award of MCA degree course in the Department of Computer Applications **Software Testing** LAB during the Academic year- 2022-23.

**LAB INCHARGE HEAD OF DEPARTMENT**

*Submitted for the university examination held on\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

**INTERNAL EXAMINER-I INTERNAL EXAMINER-II**

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No** | **Program Name** | **Page no** | **Date.** | **Teacher Signature** |
| **1** | **Understanding the Test cases, Test scenarios, and Test Reports.** | **5** |  |  |
| **2** | **Design the test case for the Arithmetic Calculations.** | **6-7** |  |  |
| **3** | **Design the test cases for sorting the n numbers.** | **8** |  |  |
| **4** | **Develop a Login form and prepare a test Report.** | **9** |  |  |
| **5** | **Develop a Students Mark sheet application and prepare test case reports.** | **10** |  |  |
| **6** | **Develop a test case for the triangle programs** | **11** |  |  |
| **7** | **Develop a test case for the Binary Search.** | **12** |  |  |
| **8** | **Develop an Employee Salary Processing application and prepare a test case report:** | **13** |  |  |
| **9** | **Develop a railway reservation system and prepare test case reports.** | **14** |  |  |
| **10** | **Perform the record and play using Selenium IDE** | **15** |  |  |
| **11** | **Perform validation of elements using the Assert Mechanism in Selenium IDE** | **16** |  |  |
| **12** | **Perform validation of elements using verify mechanism in selenium IDE** | **17** |  |  |
| **13** | **Understanding of working with selenium component(locator) and Selenium Web Driver.** | **18-21** |  |  |
| **14** | **Perform the following automation using Selenium WebDriver:**   * Launch a new Chrome browser. * Open Shop.DemoQA.com * Get Page Title name and Title length * Print Page Title and Title length on the Eclipse Console. * Get Page URL and verify if it is a correct page opened * Close the Browser**.** | **22-23** |  |  |
| **15** | **perform the following automation using Selenium WebDriver:**   * Firstly, open the browser. * Secondly, navigate to the [**ToolsQA Demo Website.**](https://demoqa.com/login) * Thirdly, maximize the browser window. * After that, retrieve the title of the page. * Fifthly, log in to the Website by specifying credentials. | **24-25** |  |  |

**Q1: Understanding of Test Case, Test Scenario and Test report.**

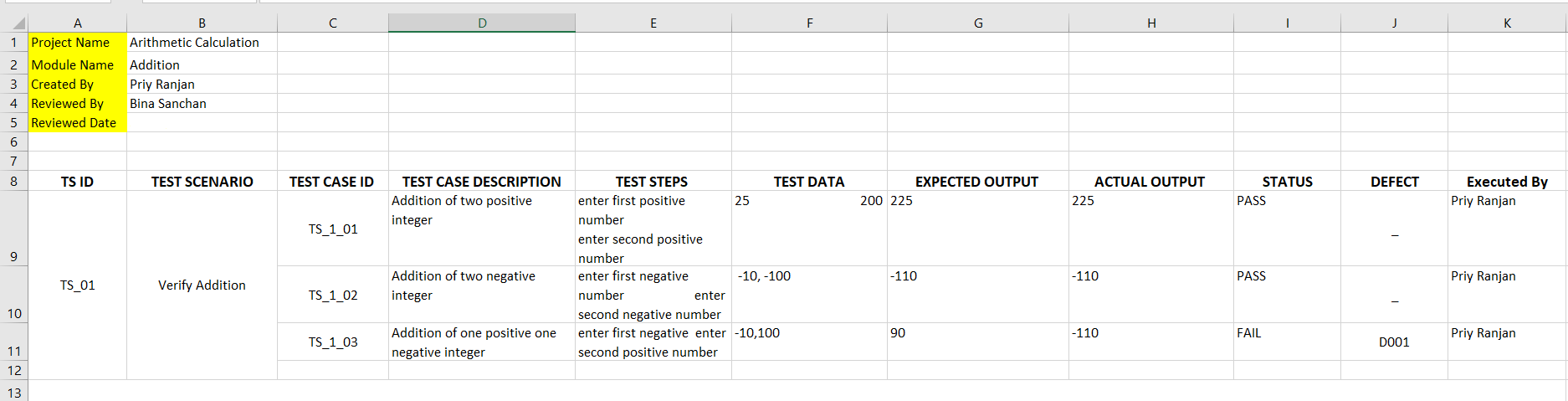
In software testing, test Case,test scenario and test report are important terms that are used to describe different aspects of the testing process.

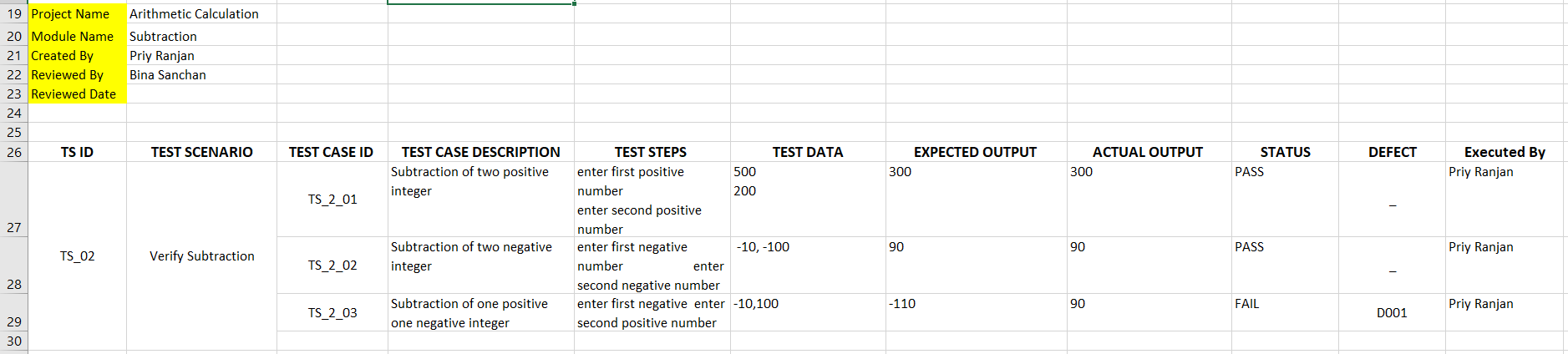
**Test Case:-** The test case is defined as a group of conditions under which a tester determines whether a software application is working as per the customer's requirements or not. Test case designing includes preconditions, case name, input conditions, and expected result.

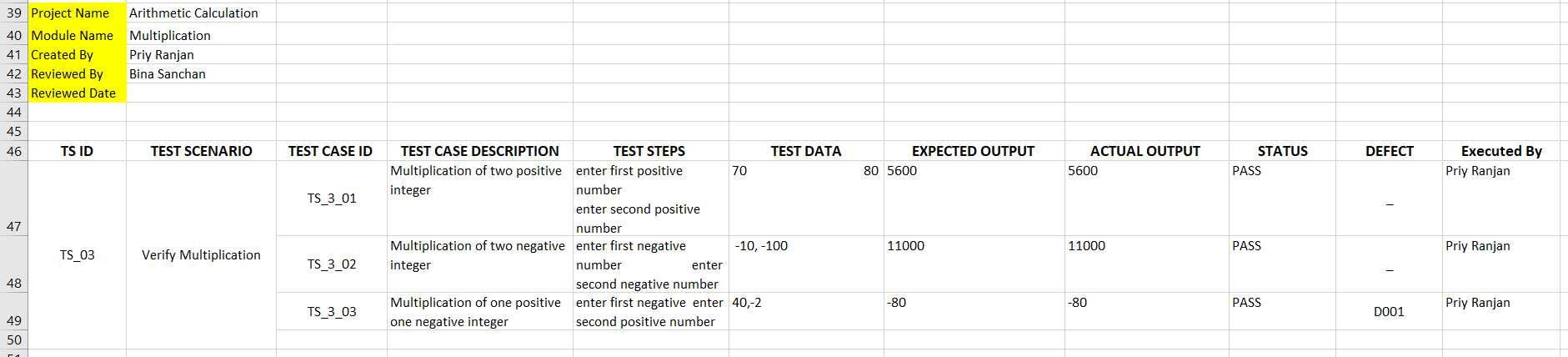
**Test Scenario:-** The test scenario is a detailed document of test cases that cover end to end functionality of a software application in liner statements. The test scenario is a high-level classification of testable requirements. These requirements are grouped on the basis of the functionality of a module and obtained from the use cases.

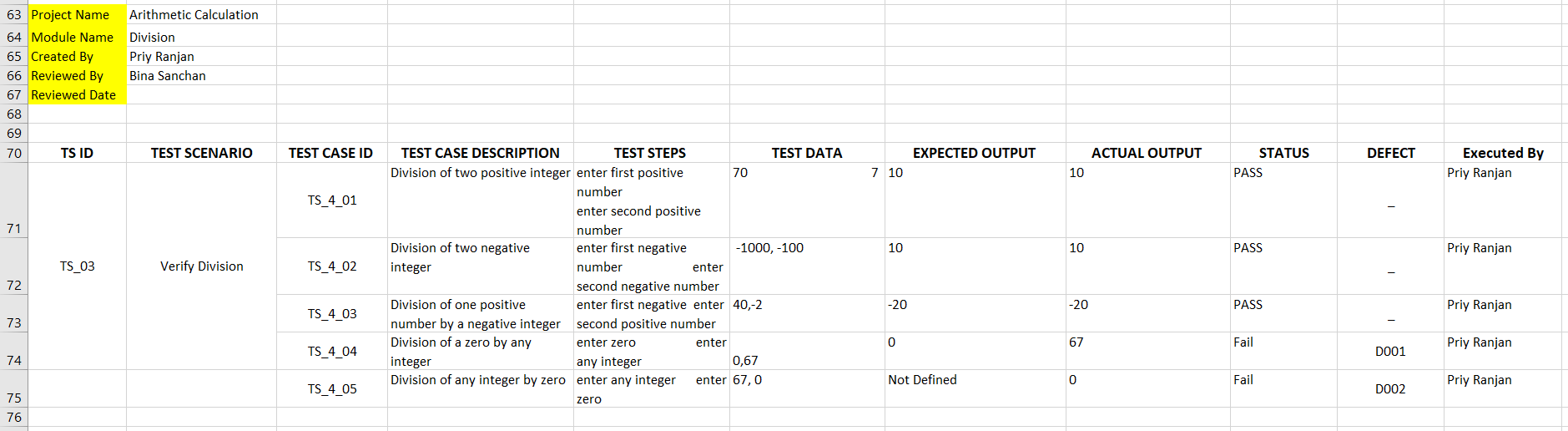
**Test report:-** **Test Report** is a document which contains a summary of all test activities and final test results of a testing project. Test report is an assessment of how well the[Testing](https://www.guru99.com/software-testing.html)is performed. Based on the test report, stakeholders can evaluate the quality of the tested product and make a decision on the software release.

**Q 2: Design the test case for the Arithmetic Calculations.**

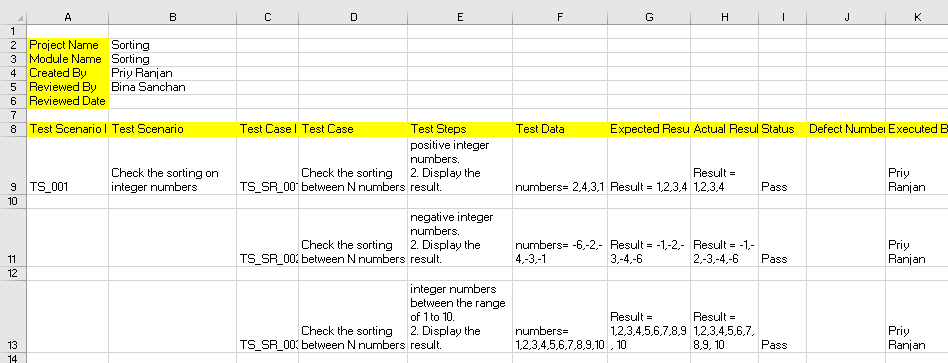


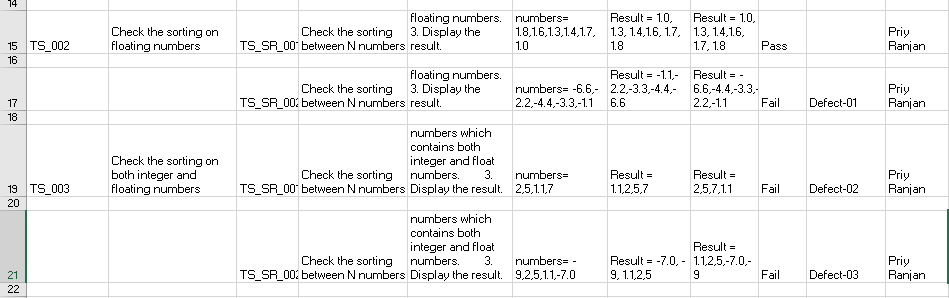




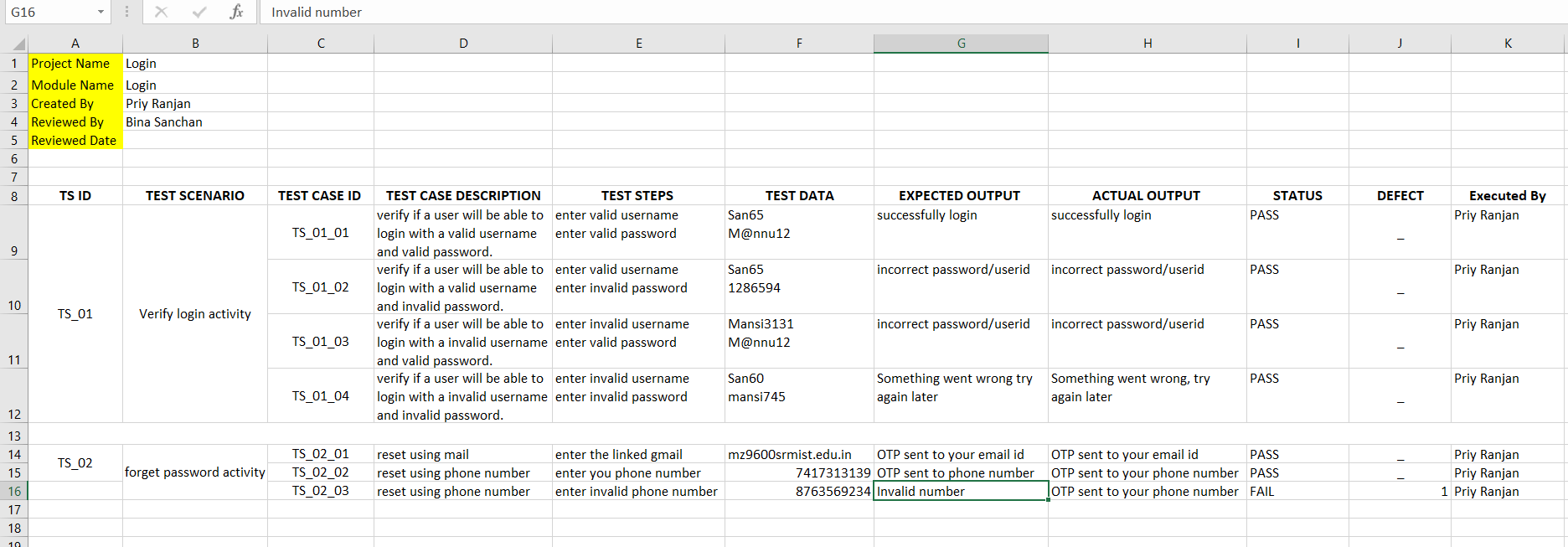


**Q 3: Design the test case report for the sorting of n numbers.**

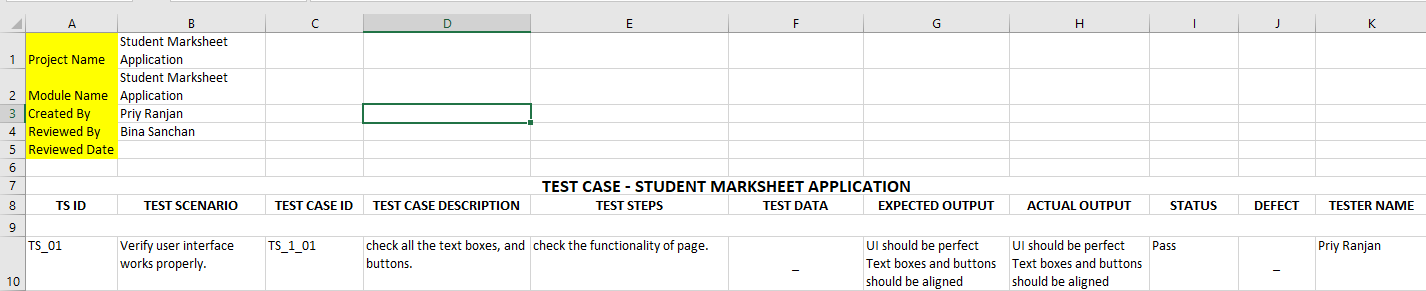


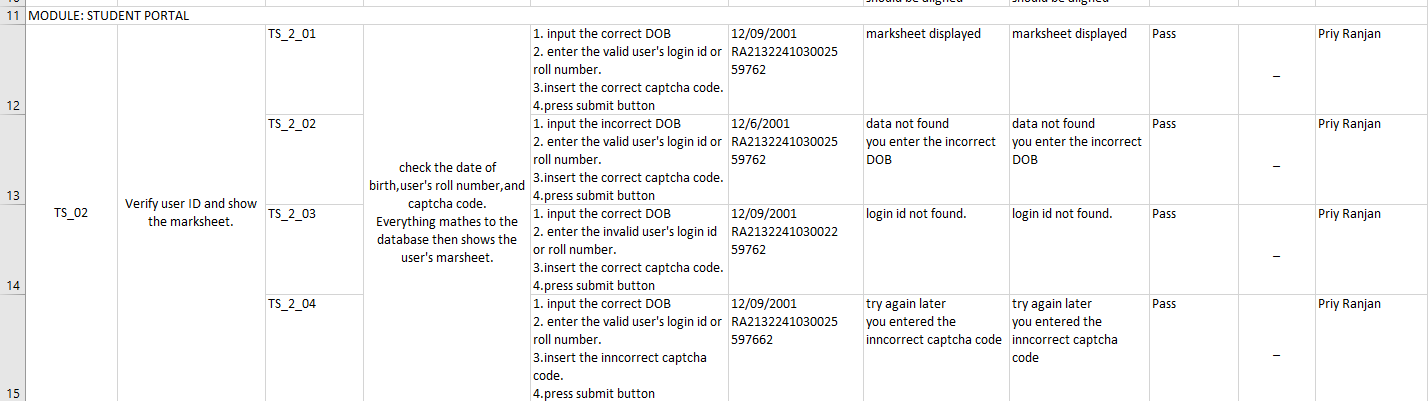


**Q4: Develop a Login form and prepare test case report.**

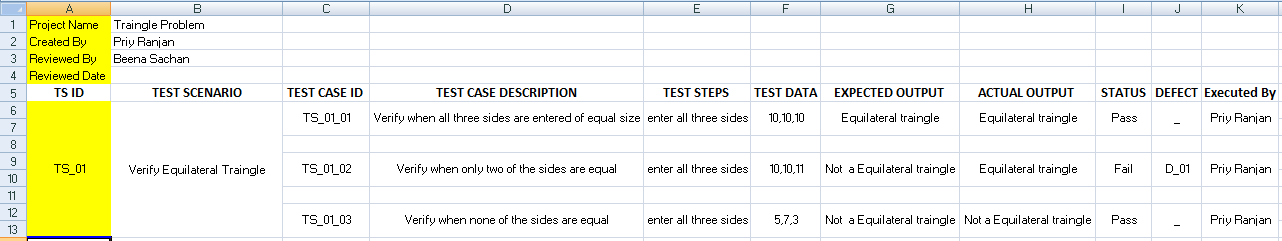


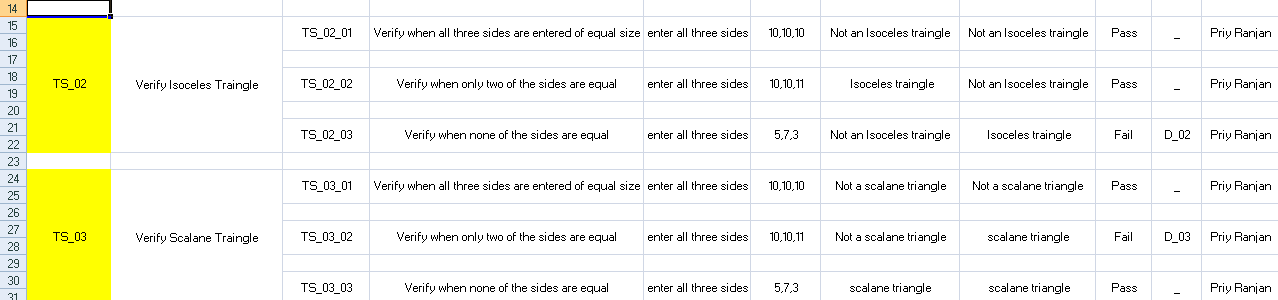
**Q5: Design the test case report for Student mark sheet application.**



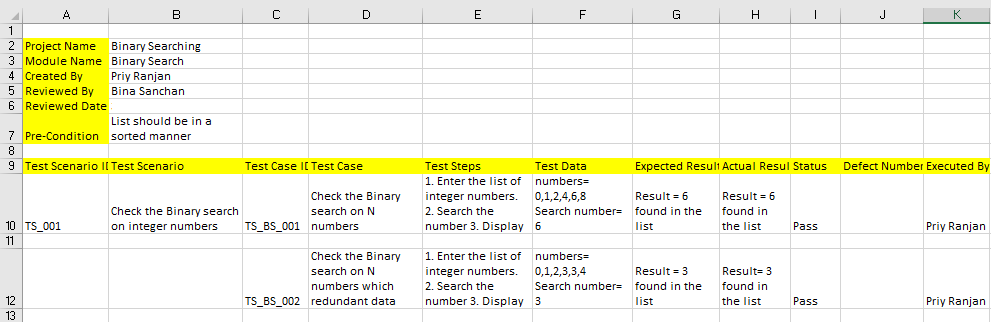


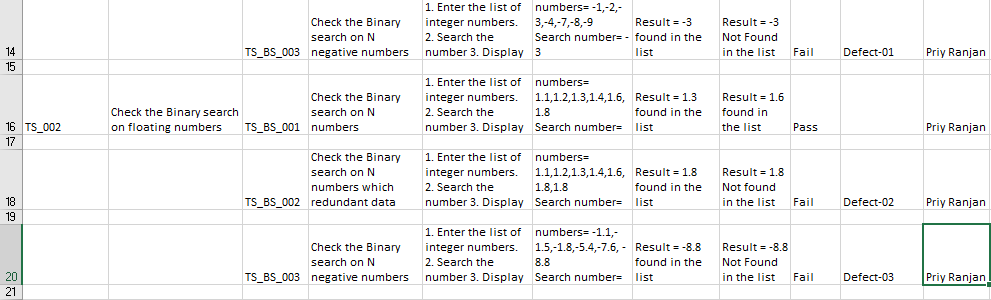
**Q6. Design test case for the triangle problem.**



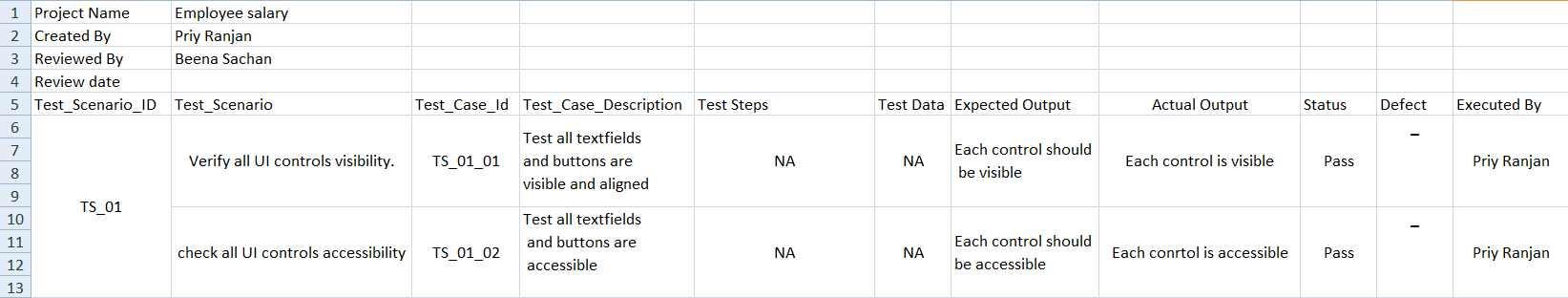


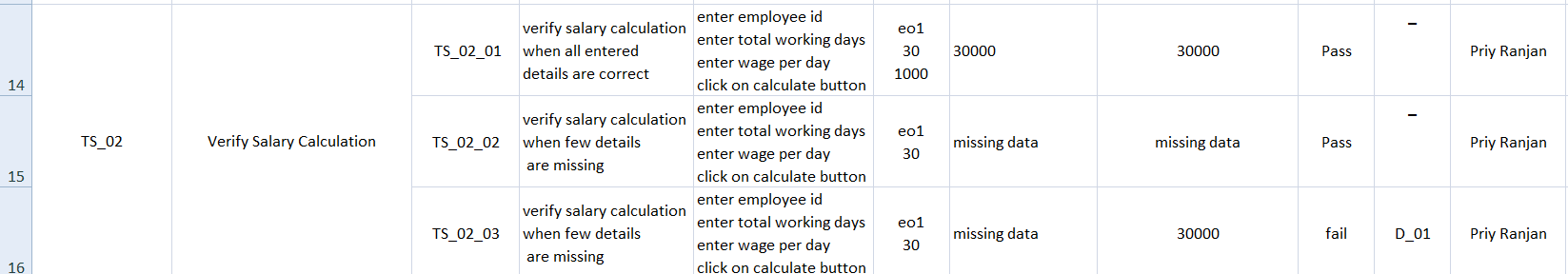
**Q7: Design the test case for Binary Search.**



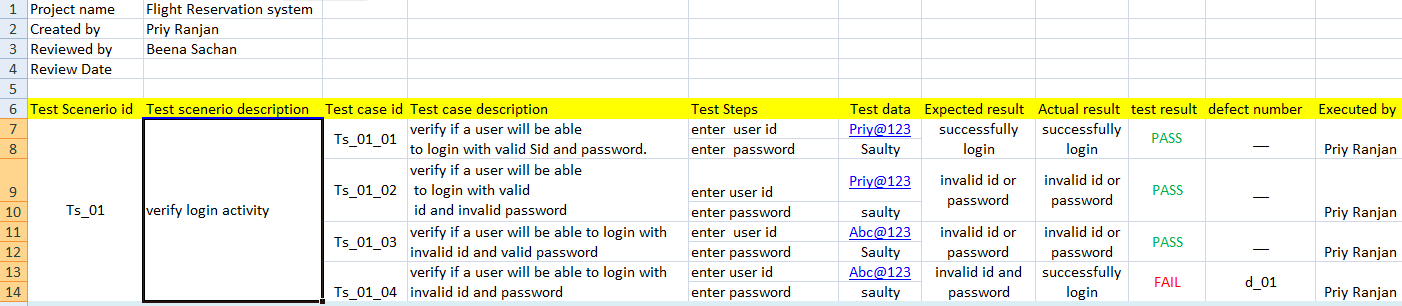


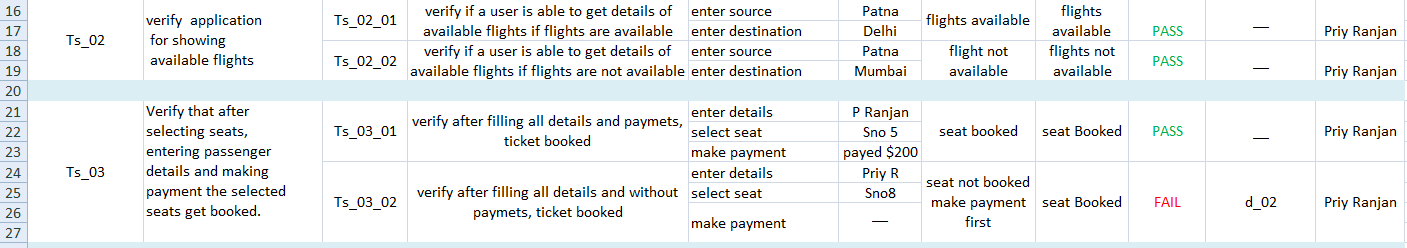
**Q8: Develop a Employee salary processing application and prepare test case report.**

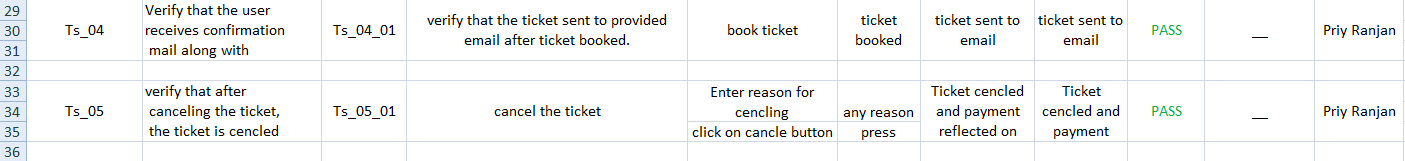




**Q9: Develop a flight Reservation System and prepare test case report.**







**Q10: Perform the record and playback using Selenium IDE**

Prerequisite - Install Selenium IDE   
  
Recording Process:  
  
Launch Firefox and open Selenium IDE from Tools Menu as shown below:

1. Ensure that the Selenium IDE is launched as shown below:
2. After the Selenium IDE is launched, ensure that the Recording Button is turned on by default. (i.e. Selenium IDE is in recording mode by default on launching).
3. While the recording button is enabled, open any website say [www.google.com](https://www.google.com/) in the Firefox.
4. In the 'Google' page, click on 'I am feeling lucky' button.
5. Turnoff the 'Recording' button on the Selenium IDE to stop recording
6. Ensure that the recorded commands are displayed under the table tab.

Playback Process:

1. Open any other site say [www.yahoo.com](http://www.yahoo.com/) in the Firefox browser and Click on the 'Play current test case' button on the Selenium IDE.
2. On clicking the 'Play current test case' button, observe that the recorded commands are executed automatically in the Firefox Browser.

**Q11. Perform validation of elements using Assert mechanism in Selenium IDE.**

The following steps will show how verify mechanism validates elements:

1. Launch Selenium IDE from Firefox Browser -> Tools Menu  
2. Ensure that the Selenium IDE is launched and also ensure that 'Record' option is enabled by default.  
3. Type [www.google.com](https://www.google.com/) in the Firfox Address bar and press 'Enter' button on the keyboard.  
4. Ensure that the Google page is displayed in the Firefox Browser.

5. In Google Page, right click on the UI element to be validated i.e Google Logo.

6. In the right click menu options, select 'Show All Available commands' option.

7. Ensure that the sub-menu option for the 'Show All Available commands' option is displayed and select 'AssertElementPresent' option .

8. In Selenium IDE -> Table tab, ensure that assertElementPresent element is added .:

9. Click on the 'Record' button to turn off the recording process as shown below:

10. Playback or Run the above recorded validation, by click on the 'Play current test case' option on the Selenium IDE .

11. After the test Run, ensure that the 'assertElementPresent' command under the Selenium IDE -> Table tab is displayed in Green color (i.e.Green means the step got passed and the UI element is  present i.e. Google Logo in this example is present on the Google Page).

12. If the element is not present (i.e. Google Logo in this example), then the step will  fail and displayed in red color as shown below:

13. If in case the test fails as shown in the step 12, all the steps to be executed after the failed step wont be executed.

**Q12. Perform validation of elements using Verify mechanism in Selenium IDE.**

The following steps will show how verify mechanism validates elements:  
  
1. Launch Selenium IDE from Firefox Browser -> Tools Menu  
2. Ensure that the Selenium IDE is launched and also ensure that 'Record' option is enabled by default.  
3. Type www.google.com in the Firfox Address bar and press 'Enter' button on the keyboard.  
4. Ensure that the Google page is displayed in the Firefox Browser  
5. In Google Page, right click on the UI element to be validated i.e Google Logo.

|  |
| --- |
|  |
|  |

6. In the right click menu options, select 'Show All Available commands' option.

|  |
| --- |
|  |
|  |

7. Ensure that the sub-menu options for the 'Show All Available commands' option is displayed and select 'VerifyElementPresent' option.  
8. In Selenium IDE -> Table tab, ensure that verifyElementPresent element is added .

|  |
| --- |
|  |
|  |

9. Click on the 'Record' button to turn off the recording process .

|  |
| --- |
|  |
|  |

10. Playback or Run the above recorded validation, by click on the 'Play current test case' option on the Selenium IDE.

|  |
| --- |
|  |
|  |
| 11. After the test Run, ensure that the 'verifyElementPresent' command under the Selenium IDE -> Table tab is displayed in Green color (i.e.Green means the step got passed and the UI element is  present. |
|  |

12. If the element is not present (i.e. Google Logo in this example), then the step will  fail and displayed in red color.

|  |
| --- |
|  |
|  |

13. If in case the test fails as shown in the step 12, all the steps to be executed after the failed step will also get executed.

**Q13: Understanding of working with selenium component(locator) and Selenium Webdriver.**

**Locator**

**Locators** are one of the essential components of Selenium infrastructure, which help Selenium scripts in **uniquely identifying the WebElements**(such as text box, button, etc.)

In Selenium Automation, Locators are used to locate the UI (User Interface) elements of a page like Text Box field, Button etc. Different Types of Locators to identify UI elements. Selenium uses different types of locators to identify the UI elements on the page.

The following is the list of locators based on the priority and recommendations to use:  
1. ID  
2. Name  
3. Class Name  
4. Link  
5. CSS  
6. XPath  
  
Practically, most of the people will try to locate the UI element using a unique ID. If  unique ID value is not available, they will use CSS selector locator followed by XPath locator.

**Why different types of locators are used instead of having only one kind of locator to identify the UI elements ?**While building the application code, it is seen as a good practice to make sure that every element you need to interact with has an ID attribute and a Name attribute. Unfortunately, developers may not have provided ID or Name attributes to some elements in your application. So identifying those kind of elements may not be possible if we don't use the other type of locators like XPath or CSS selector locators.

package TestPackage;

import java.util.List;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

public class Selenium {

public static void main(String[] args) {

String exePath = "C:\\Selenium\\chromedriver\\chromedriver.exe";

System.setProperty("webdriver.chrome.driver", exePath);

WebDriver driver = new ChromeDriver();

driver.get("https:\\demoqa.com\\");

/\*\*Locate by ID Attribute

\* URL - https://demoqa.com/automation-practice-form

\*/

driver.get("https://demoqa.com/automation-practice-form");

driver.findElement(By.id("firstName"));

/\*\*

\* Locate by Name attribute

\* URL - https://demoqa.com/automation-practice-form

\*/

driver.get("https://demoqa.com/automation-practice-form");

driver.findElement(By.name("gender"));

/\*\*

\* Locate by cssSelector attribute

\* URL - https://demoqa.com/text-box

\*/

driver.get("https://demoqa.com/text-box");

driver.findElement(By.cssSelector("input[id= ‘userName’]"));

/\*\*

\* Locate by xpath attribute

\* URL - https://demoqa.com/text-box

\*/

driver.get("https://demoqa.com/text-box");

driver.findElement(By.xpath("//input[@id='userName']"));

}}

**WebDriver**

Selenium WebDriver is a set of open-source [APIs,](https://en.wikipedia.org/wiki/Application_programming_interface) which provided the capabilities to interact with any of the modern web-browsers and then, in-turn to automate the user actions with that browser. It is an essential component of the Selenium family. As we know, Selenium is not an independent tool; rather, it is a collection of tools that make the Selenium suite, which was created when two projects Selenium RC and WebDriver were merged.

**How Selenium WebDriver works?**

When a user writes a WebDriver code in Selenium and executes it, the following actions happen in the background –

An HTTP request generates, and it goes to the respective browser driver (Chrome, IE, Firefox). There is an individual request for each Selenium command.

The browser driver receives the request through an HTTP server.

The HTTP server decides which actions/instructions need to execute on the browser.

The browser executes the instructions/steps as decided above.

The HTTP server then receives the execution status and then sends back the status to an automation script, which then shows the result ( as passed or an exception or error).

**Basic Steps in a Selenium WebDriver Script**

1. Create a WebDriver instance.
2. Navigate to a webpage.
3. Locate a web element on the webpage via [locators in selenium](https://www.browserstack.com/guide/locators-in-selenium).
4. Perform one or more user actions on the element.
5. Preload the expected output/browser response to the action.
6. Run test.
7. Record results and compare results from them to the expected output.

**How Selenium WebDriver Works**

On a high-level, Selenium WebDriver works in three steps:

Test commands are converted into an HTTP request by the JSON wire protocol.

Before executing any test cases, every browser has its own driver, which initializes the server.

The browser then starts receiving the request through its driver.

Let’s take an example with the code snippet below:

WebDriver driver = new ChromeDriver ();

driver. get (<https://www.browserstack.com>)

**Q14: perform the following automation using Selenium WebDriver:**

* **Launch a new Chrome browser.**
* **Open Shop.DemoQA.com**
* **Get Page Title name and Title length**
* **Print Page Title and Title length on the Eclipse Console.**
* **Get Page URL and verify if it is a correct page opened**
* **Close the Browser.**

**solution**

package automationFramework;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class WebDriverCommands {

public static void main(String[] args) {

String driverExecutablePath = "D:\\Drivers\\chromedriver.exe";

System.setProperty("webdriver.chrome.driver", driverExecutablePath);

// Create a new instance of the FireFox driver

WebDriver driver = new ChromeDriver();

// Storing the Application Url in the String variable

String url = "https://www.shop.demoqa.com";

//Launch the ToolsQA WebSite

driver.get(url);

// Storing Title name in the String variable

String title = driver.getTitle();

// Storing Title length in the Int variable

int titleLength = driver.getTitle().length();

**// Printing Title & Title length in the Console window**

System.out.println("Title of the page is : " + title);

System.out.println("Length of the title is : "+ titleLength);

// Storing URL in String variable

String actualUrl = driver.getCurrentUrl();

if (actualUrl.equals(url)){

System.out.println("Verification Successful - The correct Url is opened.");

}else {

System.out.println("Verification Failed - An incorrect Url is opened.");

//In case of Fail, you like to print the actual and expected URL for the record purpose.

System.out.println("Actual URL is : " + actualUrl);

System.out.println("Expected URL is : " + url);

}

**//Closing browser**

driver.close();

}}

s**Q15: perform the following automation using Selenium WebDriver:**

* **Firstly, open the browser.**
* **Secondly, navigate to the**[**ToolsQA Demo Website.**](https://demoqa.com/login)
* **Thirdly, maximize the browser window.**
* **After that, retrieve the title of the page.**
* **Fifthly, log in to the Website by specifying credentials.**

**Solution:**

package firstPackage;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

public class MyFirstTestClass {

public static void main(String[] args){

//Setting the driver path

System.setProperty("webdriver.chrome.driver","E:\\Softwares\\chromedriver.exe");

//Creating WebDriver instance

WebDriver driver = new ChromeDriver();

//Navigate to web page

driver.get("https://demoqa.com/login");

//Maximizing window

driver.manage().window().maximize();

//Retrieving web page title

String title = driver.getTitle();

System.out.println("The page title is : " +title);

//Locating web element

WebElement uName = driver.findElement(By.id(“userName"));

WebElement pswd = driver.findElement(By.id('password"));

WebElement loginBtn = driver.findElement(By.id(“login”));

//Peforming actions on web elements

uName.sendKeys("testuser");

pswd.sendKeys("Password@123");

loginBtn.click();

//Closing browser session

driver.quit();

}

}