Index:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1.Locators |  |  |  |  |  |
| 2.inbuilt methods |  |  |  |  |  |
| 3.close and quit |  |  |  |  |  |
| 4.navigation commands |  |  |  |  |  |
| 5.visual studio setup c# |  |  |  |  |  |
| 6.xpath |  |  |  |  |  |
| 10.assert |  |  |  |  |  |
| 11.screenshot |  |  |  |  |  |

1. **What is meant by a locator and name a few different types of locators present in Selenium**

A locator is an address for uniquely identifying web elements within a web page. There are different types of locators present in Selenium to identify web elements uniquely.

1. ID
2. ClassName
3. Name
4. Tagname
5. Linktex
6. Partial Linktext
7. Xpath
8. CSS selector
9. **Methods in Selenium that provide information about the current browser window or allow you to interact with it. Here are some commonly used ones:**
10. **Getting Current URL, Title, and Page Source:**

**Retrieve information about the current page.**

**string currentUrl = driver.Url;**

**string pageTitle = driver.Title;**

**string pageSource = driver.PageSource;**

1. **WindowHandle Property:**

//Gets the current window handle.

**string windowHandle = driver.CurrentWindowHandle;**

**Console.WriteLine("Current Window Handle: " + windowHandle);**

//Gets a list of all window handles if there are multiple browser windows.

**IReadOnlyCollection<string> windowHandles = driver.WindowHandles;**

**foreach (var handle in windowHandles)**

**{**

**Console.WriteLine("Window Handle: " + handle);**

**}**

1. **Manage Cookies:**

Manage().Cookies allows you to interact with cookies.

**// Get all cookies**

**var allCookies = driver.Manage().Cookies.AllCookies;**

**// Add a new cookie**

**driver.Manage().Cookies.AddCookie(new Cookie("cookieName", "cookieValue"));**

1. **Manage Timeouts:**

// Set the implicit wait time

**driver.Manage().Timeouts().ImplicitWait = TimeSpan.FromSeconds(10);**

// Set the page load timeout

**driver.Manage().Timeouts().PageLoad = TimeSpan.FromSeconds(30);**

1. **Checkboxes and Radio Buttons:**

Toggle the state of checkboxes and radio buttons.

**// Assuming 'element' is a checkbox or radio button**

**if (!element.Selected)**

**{**

**element.Click(); // Check the checkbox or select the radio button**

**}**

1. **Element State:**

Check if an element is enabled, displayed, or selected.

**// Check if an element is displayed**

**bool isDisplayed = element.Displayed;**

**// Check if an element is enabled**

**bool isEnabled = element.Enabled;**

**// Check if a checkbox or radio button is selected**

**bool isSelected = element.Selected;**

1. **Mouse and Keyboard Actions:**

Perform mouse and keyboard actions.

// Mouse actions

**Actions actions = new Actions(driver); actions.MoveToElement(element).Click().Perform();**

// Keyboard actions

**actions.SendKeys(Keys.Control + "a").Perform();**

1. **Mouse Actions:**

Perform mouse actions using the Actions class.

// Move to an element

**Actions actions = new Actions(driver);**

**actions.MoveToElement(element).Perform();**

// Click and hold, then release

**actions.ClickAndHold(element).Release().Perform();**

1. **Keyboard Actions:**

Simulate **keyboard actions using the Actions class.**

**// Send keys to an element**

**actions.SendKeys(element, Keys.Control + "a").Perform();**

1. **Implicit and Explicit Waits:**

Implicit wait sets a maximum time for the driver to wait when trying to find an element if it is not immediately available. Explicit wait is more specific, allowing you to wait for a certain condition to be true before proceeding.

**// Implicit Wait**

**driver.Manage().Timeouts().ImplicitWait = TimeSpan.FromSeconds(10);**

**// Explicit Wait**

**WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(10));**

**IWebElement element = wait.Until(ExpectedConditions.ElementIsVisible(By.Id("elementId")));**

1. **File Uploads:**

Interact with file upload inputs.

// Find file input element and send keys with file path

**IWebElement fileInput = driver.FindElement(By.Id("fileInput"));**

**fileInput.SendKeys("C:\\path\\to\\file.txt");**

1. **Drag and Drop:**

**Perform drag-and-drop operations using the Actions class.**

**IWebElement sourceElement = driver.FindElement(By.Id("sourceElement"));**

**IWebElement targetElement = driver.FindElement(By.Id("targetElement"));**

**Actions actions = new Actions(driver);**

**actions.DragAndDrop(sourceElement, targetElement).Perform();**

1. **Touch Actions (Mobile Automation):**

**Simulate touch gestures for mobile automation.**

**TouchActions touchActions = new TouchActions(driver);**

**touchActions.SingleTap(element).Perform();**

1. **Browser Resizing:**

**Change the size of the browser window during a test.**

**// Resize the browser window**

**driver.Manage().Window.Size = new Size(800, 600);**

1. **Window Position:**

**Set the position of the browser window on the screen.**

**// Set the position of the browser window**

**driver.Manage().Window.Position = new Point(100, 100);**

1. **Action Chains:**

**Perform a sequence of actions as a chain, useful for complex interactions.**

**// Example of action chain: Hover over an element and click**

**Actions actions = new Actions(driver);**

**actions.MoveToElement(elementToHover).Click().Perform();**

1. **Element Attributes:**

Retrieve and manipulate attributes of HTML elements.

**// Get attribute value**

**string attributeValue = element.GetAttribute("attributeName");**

**// Set attribute value (using JavaScript)**

**((IJavaScriptExecutor)driver).ExecuteScript("arguments[0].setAttribute('attributeName', 'value')", element);**

1. **What is the major difference between driver.close() and driver.quit()?**

**driver.close() :**

This command closes the browser’s current window. If multiple windows are open, the current window of focus will be closed.

**driver.quit() :**

When quit() is called on the driver instance and there are one or more browser windows open, it closes all the open browser windows.

1. **Different types of navigation commands.**

>> We have four Navigation commands

* **GoToUrl** driver.Navigate().GoToUrl("https://toolsqa.com");
* **Back** driver.Navigate().Back();
* **Forward** driver.Navigate().Forward();
* **Refresh** driver.Navigate().Forward();

1. **How to initialize a webdriver in C# selenium and how it is different from java selenium.**

In Java, the System.setProperty("webdriver.chrome.driver", "path"); line is used to set the system property for the ChromeDriver executable.

* However, in C# with Selenium, the approach is different. In C#, you use ChromeOptions to set properties for the ChromeDriver.

**// Set the path to the ChromeDriver executable**

**string chromeDriverPath = @"C:\Path\To\Your\ChromeDriver.exe";**

**// Set Chrome options with desired properties**

**ChromeOptions options = new ChromeOptions();**

**// Initialize ChromeDriver with the specified options**

**IWebDriver driver = new ChromeDriver(chromeDriverPath, options);**

1. **How to create a Selenium C# project in Visual Studio**
2. After installation of Visual Studio Click on GetSatrted
3. Right-click on the **File** menu, go to **New** and select **Project** option
4. Then, select **Windows** platform options from **All platforms** drop-down list.
5. And search for the .net Framework templatein the **search field,** and select **Console App (.NET Framework)** from it.
6. After that, click on the **Next** button
7. Once we clicked on the next button, **Configure your new project** window will appear on the screen, where we will provide our **Project name [SeleniumTest],** and clicking on the **Create** button

**Add References in Visual Studio:**

1. Once the project creation is done, we will add the References of **Selenium WebDriver** and **Chrome driver** with the help of the **NuGet Package Manager**in the Visual Studio.
2. In the **Solution Explorer**, right-click on the **References** and select **Manage NuGet Packages** option.
3. **XPath Syntax:**

Xpath = //tagname[@attribute=’value’]

1. **Writing XPath using different methods.**

**Contains() :**

Syntax : //tagname[contains(@attribute,’value’]

Xpath=//\*[contains(@name,'btn')]

Xpath=//\*[contains(@id,'message')]

**Using OR & AND :**

Syntax : //tagname[@attribute=’value’ or @attribute = ‘value’]

Xpath=//\*[@type='submit' or @name='btnReset']

Syntax : //tagname[@attribute=’value’ and @attribute = ‘value’]

Xpath=//input[@type='submit' and @name='btnLogin']

**Starts-with :**

Syntax : //tagname[starts-with(@attribute,’value’)]

Xpath=//label[starts-with(@id,'message')]

**XPath Text() Function:**

Syntax : //tagname[text()=’value’]

Xpath=//td[text()='UserID']

Where …td= table data

value = text containing

1. **Writing xapth using Xpath axes methods:**

**Using Following:**

Syntax : //tagname[@attribute=’value’]following::tagname[@attribute=’value’]

//td[@align='right']//following::label[@id='message23']

**Using Ancestor:**

Syntax: //tagname[@attribut=’value’]//ancestor::tagname[@attribute=’value’]

//td[@align='right']//ancestor::table[@align='center']

Syntax: //tagname[text()=’’text inside that’]//ancestor::tagname[@attribute=’value’]

Xpath=//\*[text()='Enterprise Testing']//ancestor::div[1]

**Using Child:**

Xpath=//\*[@id='java\_technologies']//child::li[1]

//\*[@id='java\_technologies']//child::li//a[@title='SoapUI']

**Using preceding:** Xpath=//\*[@type='submit']//preceding::input

**Using Following sibling:** Xpath=//\*[@type='submit']//following-sibling::input

**Using Parent:** Xpath=//\*[@id='rt-feature']//parent::div

**Using Self:** Xpath =//\*[@type='password']//self::input

**Using Descendent:** Xpath=//\*[@id='rt-feature']//descendant::a

1. **How to assert the title of a webpage?**

Assert.assertEquals(actualTitle, expectedTitle);

String actualTitle = driver.getTitle(); // Get the title of the webpage and store in a variable

String expectedTitle = “abcdefgh"; // Type in the expected title

if(actualTitle.equalsIgnoreCase(expectedTitle))

System.out.println("Title Matched");

else

System.out.println("Title didn't match");

**Assert.assertEquals(actualTitle, expectedTitle);**

1. **Screenshots in Webdriver**

TakeScreenshot interface can be used to take screenshots in WebDriver.

getScreenshotAs() method can be used to save the screenshot

File scrFile = ((TakeScreenshot)driver).getScreenshotAs(outputType.FILE);

1. **Explain the difference between FindElement and FindElements methods in Selenium WebDriver.**

**FindElement :** returns the first matching element on the page, while

**FindElements :** returns a collection of all matching elements. If no element is found,

FindElement throws an exception, while FindElements returns an empty list**.**

1. **How to wait until a web page has been loaded completely in Selenium?**

One approach is to use the "implicit wait" command in Selenium, which instructs the web driver to wait a certain amount of time before throwing an error if the element is not found or loaded. Another option is to use the "explicit wait" command to wait for a specific element to appear on the page before proceeding with the script.

1. **How do you handle file uploads in Selenium WebDriver with C#?**

To handle file uploads in Selenium WebDriver with C#, you can use the SendKeys() method to enter the file path into the file input element. For example:

**IWebElement fileInput = driver.FindElement(By.Id("fileInput"));**

**fileInput.SendKeys("path/to/file.txt");**

1. **What are the types of waits supported by WebDriver?**

**Implicit Wait :**

Implicit wait pauses the execution of the web driver for a specified period before throwing any error. The specified time is based upon the time required by the web elements to get ready for the test, and hence get loaded on the page. However, the execution time of the overall test increases.

If the particular element takes more time than what is specified, the Selenium web driver throws an error “NoSuchElementException”.

**The syntax for using the Implicit wait command in Selenium C# is as follows**.

driver.Manage().Timeouts().ImplicitWait = TimeSpan.FromSeconds(Value);

driver.Manage().Timeouts().ImplicitWait = TimeSpan.FromSeconds(Value);

**Explicit Wait:**

In the Implicit Wait command, it waits for a specific period. However, in Explicit Wait, it will wait till certain conditions occur. The wait here for a web element is not for a specific period but till the web element is ready in the DOM for testing. This is the reason it is also known as “smart wait”.

The Explicit Wait command checks the condition (element to become clickable, displayed, etc) every 250ms. Moreover, Implicit wait is only applicable with FindElement methods, however, Explicit Wait has several possible conditions.

The Selenium Webdriver provides two classes for the implementation of Explicit Wait.

* WebDriverWait
* ExpectedConditions

The WebDriverWait calls the ExpectedConditions class method until it returns successfully or the specified time is over. It is a major improvement for Implicit Wait, as there is no extra delay in the test.

**The syntax for the usage of Explicit Wait is as follows:**

WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(10));

wait.Until(ExpectedConditions.ElementExists(By.Id("id")));

WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(10));

wait.Until(ExpectedConditions.ElementExists(By.Id("id")));

**Fluent Wait Command in C#:**

The Fluent Wait command in Selenium is similar to Explicit Wait in so many aspects. It allows you to control the Web Driver to set the specified time for some condition to appear before it could throw the error “ElementNotVisibleException”.

The main advantage of implementing the Fluent Wait command is setting the polling frequency. Polling frequency is the frequency at which the driver checks for the element whether it has loaded or not. It has the attribute .pollingfrquency, and its default value is 500ms, which means the driver will check every 500 milliseconds before throwing the error.

**The syntax for the usage of Fluent Wait is as follows:**

**DefaultWait<IWebDriver> fluentWait = new DefaultWait<IWebDriver>(driver);**

**fluentWait.Timeout = TimeSpan.FromSeconds(5);**

**fluentWait.PollingInterval = TimeSpan.FromMilliseconds(polling\_interval\_in\_ms);**

DefaultWait<IWebDriver> fluentWait = new DefaultWait<IWebDriver>(driver);

fluentWait.Timeout = TimeSpan.FromSeconds(5);

fluentWait.PollingInterval = TimeSpan.FromMilliseconds(polling\_interval\_in\_ms);

1. **Basic Elements Operations**

* Click : driver.Click();
* SendKeys : driver.SendKeys(“input text”);
* Clear(); : driver.Clear();
* IWebElement element = driver.**FindElement**(By.**Id**(**"id"**));
* element.**Click**();
* element.**SendKeys**(**"someText"**);
* element.**Clear**();
* element.**Submit**();

1. **Select methods in C# Selenium**

IWebElement element = driver.FindElement(By.Id("the web element id"));

* SelectElement select = new SelectElement(element);
* select.SelectByIndex(1);
* select.SelectByText("Ford");
* select.SelectByValue("ford");
* select.DeselectAll();
* select.DeselectByIndex(1);
* select.DeselectByText("Ford");
* select.DeselectByValue("ford");

1. **Dropdown in Selenium C#** (Select method)

driver.Navigate().GoToUrl("https://www.example.com");

IWebElement dropdown = driver.FindElement(By.Id("dropdown"));

SelectElement select = new SelectElement(dropdown);

select.SelectByValue("option2");

// Option 2: Using FindElement and Click

// Click the dropdown to open it

dropdown.Click();

// Find and click the desired option

driver.FindElement(By.CssSelector("#dropdown option[value='option3']")).Click();

// Option 3: Using JavaScriptExecutor

IJavaScriptExecutor js = (IJavaScriptExecutor)driver;

// Execute JavaScript to select an option by value

js.ExecuteScript("arguments[0].value='option1'", dropdown);

1. **If the dropdown tag is not under the <SELECT> tag but under <div >or any other how to select from dropdown?**

if the dropdown is implemented using a <div> or any other HTML element (not a <select> tag), you can still interact with it using Selenium WebDriver. In such cases, you would typically use other methods such as simulating mouse actions, sending keys, or executing JavaScript to manipulate the dropdown and select options.

Here's an example using the Actions class to interact with a custom dropdown implemented with a <div>:

driver.Url="https://demoqa.com/select-menu";

IWebElement drpdown= driver.FindElement(By.XPath("//div[text()='Select Option']"));

Actions act= new Actions(driver);

  act.Click(drpdown).Build().Perform();

IWebElement option= driver.FindElement(By.XPath("//div[text()='Group 1, option 1']"));

    option.Click();

1. **How to handle alerts**

**IAlert a = driver.SwitchTo().Alert();**

a.getText();

a.SendKeys(“abc”);

a.Accept();

a.Dismiss();

1. **How to Handle Frames in Selenium Using WebDriver Commands?**

**By Index:**

**driver.switchTo().frame(0);**

**By Name or ID:**

**driver.switchTo().frame("frame-name");** // Switch to the frame by name

**driver.switchTo().frame("frame-id");** // Switch to the frame by ID

**By Web Element:**

**WebElement frameElement = driver.findElement(By.id("frame-id"));**// Find the frame element

**driver.switchTo().frame(frameElement);** // Switch to the frame using the web element

We must first identify the frame to handle frames in selenium using WebDriver commands. This can be done in three ways index, by name or id, and by web elements. Selenium WebDriver has a few simple steps to handle frames:

* Switch the driver's focus to the frame using the switchTo().frame() method.
* Using web driver commands, interact with the elements of the frame and perform the operations.
* Switch back to the web content by the switchTo().defaultContent() method.

1. **Handling Nested Frames in Selenium WebDriver**

driver.switchTo().frame("parent-frame"); // Switch to the parent frame

driver.switchTo().frame("child-frame"); // Switch to the child frame

// Interact with elements inside the child frame

WebElement childElement = driver.findElement(By.id("child-element"));

childElement.click();

// Switch back to the parent frame

driver.switchTo().parentFrame();

// Interact with elements inside the parent frame

WebElement parentElement = driver.findElement(By.id("parent-element"));

parentElement.click();

// Switch back to the default content

driver.switchTo().defaultContent();

1. **Drag and drop action in selenium.**

* Drag and drop operations can be performed using the Actions class

driver.SwitchTo().Frame(driver.FindElement(By.ClassName("demo-frame")));

IWebElement draggableElement = driver.FindElement(By.Id("draggable"));

IWebElement droppableElement = driver.FindElement(By.Id("droppable"));

Actions actions = new Actions(driver);

actions.DragAndDropToOffset(draggableElement, 100, 100).Perform();

**REVERSE:**

* If you want to perform a drag-and-drop operation and then revert or drag back to the original position, you can use a combination of the **Actions** class and JavaScript. Here's an example in C#:

**Point initialPosition = sourceElement.Location;**

**// Perform the drag-and-drop operation**

**Actions actions = new Actions(driver);**

**actions.DragAndDrop(sourceElement, targetElement).Perform();**

**// Perform the reverse drag to the initial position using JavaScript**

**((IJavaScriptExecutor)driver).ExecuteScript($"arguments[0].style.transform = 'translate({initialPosition.X}px, {initialPosition.Y}px)';", sourceElement);**

1. **How to handle Windows using selenium?**

* In Selenium WebDriver using C#, handling multiple browser windows or tabs involves using the WindowHandles property.

**Code:**

driver.Navigate().GoToUrl("https://www.example.com");

IWebElement newWindowLink = driver.FindElement(By.LinkText("Open New Window"));

newWindowLink.Click();

// Get the window handles

List<string> windowHandles = new List<string>(driver.WindowHandles);

// Switch to the new window

string newWindowHandle = windowHandles[1];

driver.SwitchTo().Window(newWindowHandle);

// Perform actions on the new window

// For example, navigate to a different URL in the new window

driver.Navigate().GoToUrl("https://www.example.com/newpage");

// Switch back to the main window

string mainWindowHandle = windowHandles[0];

driver.SwitchTo().Window(mainWindowHandle);

// Close the main window (or any window as needed)

driver.Close();

// Close the new window

driver.SwitchTo().Window(newWindowHandle).Close();

driver.Quit();

1. **Is there a way to type in a textbox without using SendKeys()?**

Text can be entered into a textbox using JavaScriptExecutor

JavascriptExecutor jse = (JavascriptExecutor) driver;

jse.executeScript("document.getElementById(‘email').value=“abc.efg@xyz.com”);

1. **When isEnabled(), isDisplayed(),or isSelected() was called while the element was not existing, What exception will WebDriver throw ?**

When isEnabled(), isDisplayed(),or isSelected() was called while the element was not existing, WebDriver will throw **a NoSuchElementException**.

1. **How to scroll down vertically on a web page in Selenium by defining the number of pixels?**

* A Scroll is a JavaScript method. The JavaScriptExecutor provides an interface that enables QAs to run JavaScript methods from Selenium scripts. Hence, to scroll up or down with Selenium, a JavaScriptExecutor is a must.

**IJavaScriptExecutor js = (IJavaScriptExecutor)driver;**

**js.ExecuteScript($"window.scrollBy(0, {pixels});");**

**IJavaScriptExecutor js = (IJavaScriptExecutor)driver; js.ExecuteScript($"window.scrollBy({pixels}, 0);");**

1. **How to do slide using Selenium.**

To perform sliding actions using Selenium in C#, you can use the Actions class provided by Selenium WebDriver. Here's an example code to perform a sliding action:

using OpenQA.Selenium.Interactions;

// Find the element to slide

IWebElement slider = driver.FindElement(By.XPath("//input[@type='range']"));

// Get the size of the slider

int width = slider.Size.Width;

// Create an instance of Actions class

Actions action = new Actions(driver);

// Move slider by 100 pixels

action.ClickAndHold(slider)

.MoveByOffset(100, 0) // Change the offset according to your requirement

.Release()

.Perform();

// Alternatively, you can use the DragAndDropToOffset method

// action.DragAndDropToOffset(slider, 100, 0).Perform();

// Optional: Print the slider value after sliding

Console.WriteLine("Slider value after sliding: " + slider.GetAttribute("value"));

1. **What is Cucumber? Why is it used?**

* Cucumber is a testing tool based on Behavior Driven Development (BDD) framework. It is used to run functional tests written in plain text and develop test cases for software functionality. It plays a supporting role in automated testing.
* In other words, we can say that "Cucumber is a software tool used by the testers to develop test cases for the testing of behavior of the software."

1. **What is the main aim of the Behavior Driven Development (BDD) framework?**

The main aim of the Behavior Driven Development framework is to make various project roles such as Business Analysts, Quality Assurance, Developers, etc., understand the application without diving deep into the technical aspects.

1. **What is gherkin Language?**

The Cucumber tool uses the Gherkin language, a simple English representation of the application behavior. The Gherkin language uses several keywords to describe the behavior of applications such as Feature, Scenario, Scenario Outline, Given, When, Then, etc.

1. **What are the two files required to execute a Cucumber test scenario?**

Following are the two files required to execute a Cucumber test scenario:

* Features - A feature file is used to provide a high-level description of an Application Under Test (AUT). The first line of the feature file must start with the keyword 'Feature' followed by the description of the application under test. A feature file may include multiple scenarios within the same file, and the extension of the feature file must be ".feature."
* Step Definition - In Cucumber, a step definition is the actual code implementation of the feature mentioned in the feature file.

1. **What is the starting point of execution for feature files?**

* When Cucumber is integrated with Selenium, the starting point of execution must be from the TestRunner class.

1. **What are the various keywords used in the Cucumber tool for writing a scenario?**

Following are the keywords that are used for writing a scenario in the Cucumber tool:

* Given
* When
* Then
* And
* But

1. **What do you understand by regular expressions?**

A Regular Expression (or Regex) is a pattern (or filter) that describes a set of strings that matches the pattern. In other words, a regex accepts a certain set of strings and rejects the rest.

1. **What is the usage of a Scenario Outline in the Cucumber tool?**

In Cucumber, a Scenario outline is used as a parameter of scenarios. This is used when the same scenario needs to be executed for multiple sets of data; however, the test steps remain the same. Scenario Outline must be followed by the keyword 'Examples', which specify the set of values for each parameter.

1. **What is the use of the Background keyword in Cucumber?**

In the Cucumber tool, the Background keyword is used to group multiple given statements into a single group. The keyword is mostly used when the same set of given statements are repeated in each scenario of the feature file.

1. **How do we define parameterize in cucumber**

* Create a Scenario Outline by adding the Scenario Outline keyword. Define placeholder parameters within angle brackets (<>) in the scenario steps. Provide examples in the Examples section with concrete values for each parameter.

1. **Types of manual testing:**

* Black Box
* White Box
* Integration
* Unit
* System
* Acceptance

1. **Black Box Testing**:

The testing approach where the internal logic and implementation details of the system or component under test are not known to the tester. It focuses on inputs and expected outputs.

1. **White Box Testing:**

Testing approach where the tester has knowledge of the internal logic, code, and implementation of the system or component. It aims to validate the correctness of the code's structure and execution.

1. **Integration Testing:**

Testing phase that focuses on verifying the interaction between different modules or components of a system to ensure they work together as intended.

1. **Unit Testing:**

Testing at the smallest unit level, usually individual functions or methods. It verifies that each part of the code performs as expected in isolation.

1. **System Testing:**

Comprehensive testing of the entire system to ensure that all components work together as a complete and integrated unit, meeting specified requirements.

1. **Acceptance Testing:**

Final phase of testing where the system is evaluated to determine if it satisfies the specified business requirements and is ready for deployment or acceptance by the end-users.

1. **Explain the difference between alpha testing and beta testing.**

Alpha testing is at the developer’s site before release. Potential clients conduct beta testing at their websites.

1. **What’s the difference between verification and validation?**

Verification evaluates the software at the development phase, ascertaining whether or not a product meets the expected requirements. On the other hand, validation evaluates the software after the development phase, making it sure it meets the requirements of the customer.

1. **What’s the difference between a bug and a defect?**

A bug is a fault in the software that’s detected during testing time, while a defect is a variance between expected results and actual results, detected by the developer after the product goes live.

1. **What is regression testing? What is retesting? How they differ from each other?**

Regression testing is performed to find out whether the updates or changes had caused new defects in the existing functions

**Retesting** : Developer says it is fixed. Doing testing once again

1. **What is random/monkey testing? When is it used?**

* By giving random data so that we can break the system.
* Load testing, Stress testing which comes under non-functionality, It is Timeboxed and Random testing, provide random values and check the response time,

1. **Differentiate between bug leakage and bug release**

**Bug Leakage** - When tested software is pushed into the market and the end-user discovers defects, this is known as bug leakage. These are bugs that the testing team overlooked throughout the testing phase.

**Bug Release** - When a certain version of software is launched into the market with some known bugs that are expected to be fixed in later versions, this is known as a bug release. These are low-priority issues that are highlighted in the release notes when sharing with end-users.

1. **What is defect life cycle?**

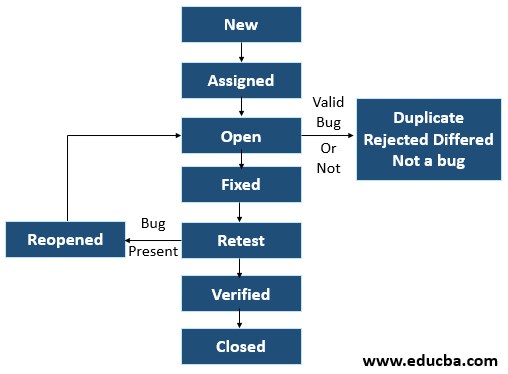
**Status**: New, Assign to Developer, Opened.

After fixing, assigns to tester to test,

Developer may Defer, Duplicate if the defect is repeated,

Developer can rejected

Tester will test the defect and close it with Closed status.



1. **Boundary Value analysis**

It is used to identify defects and errors in software by testing input values on the boundaries of the allowable ranges.

* The goal of boundary value analysis is to find any issues which may arise due to incorrect assumptions about the system behavior.

1. **Difference between priority and severity in software testing?**

Priority is the importance which is given to a bug to fix

Severity is the impact on the functionality

1. **An example of high priority and low severity bug?**

Say, a typo has high priority and low severity (Name of the bank or color of the company logo)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[**How to Find Broken Links in Selenium**](https://www.guru99.com/find-broken-links-selenium-webdriver.html)

https://www.guru99.com/find-broken-links-selenium-webdriver.html