Experiment No - 1

Q. Find out defects Old Google Homepage Test (layout inconsistencies, spelling errors, and the like) in the image.

Here are some scenarios we can consider:

1. Uneven Spacing Test Case:

- **Description:** Verify that the spacing between the "Google" logo and the search bar is consistent.
- **Steps:**

  1.Open the Google homepage.

  2.Measure the distance between the logo and the search bar.

  3.Compare it to the expected spacing (based on modern design standards).

- **Expected Result**: The spacing should be consistent and visually pleasing.

2. Button Placement Test Case:

- **Description:** Confirm the correct placement of the “I’m Feeling Lucky” button.

- **Steps:**
    1. Open the Google homepage.
    2. Locate the “I’m Feeling Lucky” button.
    3. Verify its position relative to the search bar (below or to the right).

- **Expected Result:** The button should be positioned according to the current design.

3. Spelling Error Test Case:

- **Description:** Ensure that the Google logo has the correct spelling.
- **Steps:**

    1.Open the Google homepage.

    2.Inspect the logo.

    3.Verify that it reads “Google” (not “Gooogle”).

- **Expected Result:** The logo should have the correct spelling.

4. Outdated Interface Test Case:

- **Objective:** Assess the overall design and layout for outdated elements.

- **Steps:**
    1. Compare the old Google homepage image with the current version.
    2. Look for outdated visual elements (e.g., fonts, colors, styling).

- **Expected Result:** The old version should exhibit design elements that are no longer in use.

5. Missing Features Test Case:

- **Objective:** Confirm the presence of essential features.

- **Steps:**
    1. Check if Gmail, Drive, and other Google services are prominently displayed.
    2. Compare with the current Google homepage.

- **Expected Result:** The old version should lack these features or have them less prominently displayed.

6. Browser Compatibility Test Case:

- **Objective:** Validate compatibility with different browsers.

- **Steps:**
    1. Load the old Google homepage image in various browsers (e.g., Chrome, Firefox, Edge).
    2. Observe any rendering issues or inconsistencies.
- **Expected Result:** The page should display correctly across different browsers.

7. Search Bar Design Test Case:

- **Objective:** Assess the simplicity of the search bar design.
- **Steps:**

1. Examine the search bar in the old image.
2. Compare it to the current Google search bar (with features like autocomplete).

- **Expected Result:** The old search bar should lack modern features.

Experiment No - 2

Q. Write Test Cases for a Login Page :

1. **Email/Phone Number/Username Textbox**
2. **Password Textbox**
3. **Login Button**
4. **Remember Me Checkbox**
5. **Keep Me Signed In Checkbox**
6. **Forgot Password Link**
7. **Sign up/Create an account Link**

Answer =>

1> Test Cases for a Login Page:

When creating test cases for a login page, it's important to cover a variety of scenarios to ensure the page functions correctly and securely. Here are some common test cases for a login page:

1. Successful Login

- **Test Case 1.1: Valid Credentials**
- **Description** : Verify that a user can log in with valid credentials.
- **Steps** :
    1. 1.Enter a valid username.
    2. 2.Enter the correct password for that username.
    3. 3.Click the "Login" button.
- **Expected Result** : The user is successfully logged in and redirected to the homepage or dashboard.

2. Unsuccessful Login

- **Test Case 2.1: Invalid Username**
- **Description** : Verify that an error message is displayed when an invalid username is entered.
- **Steps** :

    1.Enter an invalid username.

    2.Enter a valid password.

    3.Click the "Login" button.

- **Expected Result** : An error message is displayed indicating invalid username.

- **Test Case 2.2: Invalid Password**
- **Description**: Verify that an error message is displayed when an invalid password is entered.
- **Steps** :

    1.Enter a valid username.

    2.Enter an incorrect password.

    3.Click the "Login" button.

- **Expected Result** : An error message is displayed indicating invalid password.

Test Case 2.3: Both Fields Empty

- **Description** : Verify that an error message is displayed when both username and password fields are left empty.
- **Steps** :

    1.Leave both username and password fields empty.

    2.Click the "Login" button.

- **Expected Result** : An error message is displayed indicating that both fields are required.

3. Field Validation

Test Case 3.1: Username Field Validation

- **Description** : Verify that the username field accepts only valid characters.
- **Steps** :

  1.Enter special characters or invalid input in the username field.

  2.Click the "Login" button.

- **Expected Result** : An error message is displayed or input is rejected.

Test Case 3.2: Password Field Validation

- **Description** : Verify that the password field accepts the correct format (e.g., minimum length, required characters).
- **Steps** :

  1.Enter a password that does not meet the security criteria.

  2.Click the "Login" button.

- **Expected Result** : An error message is displayed indicating that the password does not meet the requirements.

4. UI Elements

Test Case 4.1: Presence of Login Button

- **Description** : Verify that the login button is present on the page.
- **Steps** :

  1. Load the login page.

- **Expected Result** : The "Login" button is visible and clickable.

Test Case 4.2: Presence of Forgot Password Link

- **Description**: Verify that the "Forgot Password" link is present and functional.
- **Steps** :

    1. 1.Load the login page.
    2. 2.Click the "Forgot Password" link.
- **Expected Result** : The user is redirected to the password recovery page.

5. Security

Test Case 5.1: SQL Injection

- **Description**: Verify that the login form is protected against SQL injection attacks.
- **Steps**:

    1. 1.Enter SQL injection code in the username or password fields.
    2. 2.Click the "Login" button.
- **Expected Result**: The system should reject the input and not execute any harmful commands.

Test Case 5.2: Password Masking

- **Description** : Verify that the password is masked when typed.
- **Steps** : 1. Enter a password in the password field.
- **Expected Result** : The characters should be displayed as dots or asterisks.

6. Usability

Test Case 6.1: Error Message Clarity

- **Description** : Verify that error messages are clear and helpful.
- **Steps** :

    1.Enter invalid credentials.
    2.Click the "Login" button.
- **Expected Result** : The error message should clearly indicate the problem (e.g., "Invalid username or password").

Test Case 6.2: Successful Login Redirect

- **Description** : Verify that a successful login redirects the user to the correct page.
- **Steps** :
    1.Enter valid credentials.
    2.Click the "Login" button.
- **Expected Result** : The user is redirected to the homepage, dashboard, or the page they were trying to access.

7. Compatibility

Test Case 7.1: Cross-Browser Compatibility

- **Description** : Verify that the login page functions correctly across different web browsers.
- **Steps** :
    1.Open the login page in various browsers (e.g., Chrome, Firefox, Edge)
    2.Perform login actions in each browser.
- **Expected Result** : The login page behaves consistently across all tested browsers.

Test Case 7.2: Mobile Responsiveness

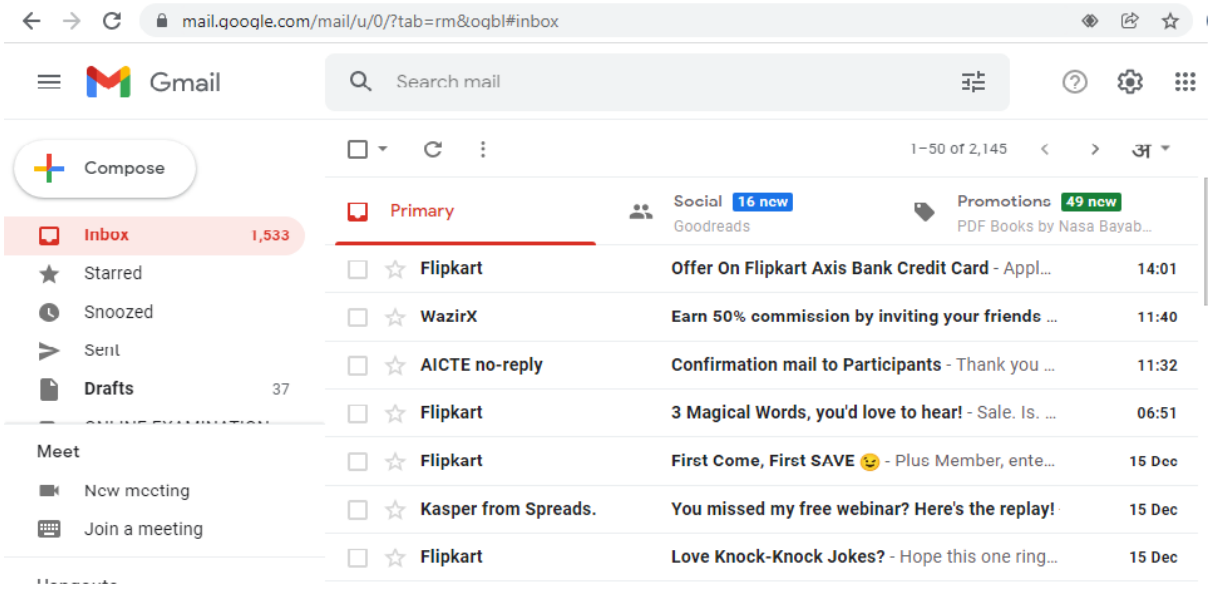
- **Description** : Verify that the login page is responsive and usable on mobile devices.
- **Steps** :
    1. 1.Open the login page on various mobile devices.
    2. 2.Perform login actions on each device.
- **Expected Result** : The login page should display correctly and be functional on all tested devices.

These test cases help ensure the login page is robust, user-friendly, and secure.

Adjust and expand them based on your specific application and requirements.

Experiment No - 3

Q. Write the Test Scenarios of Gmail Inbox Functionality.



The test scenarios for Gmail's inbox functionality. Ensuring that Gmail's inbox works seamlessly is crucial, especially considering its massive user base. Here are some test cases you can consider:

1. **Email Reception Verification:**

- **Description:** To validate the correct reception of emails in the Gmail inbox.
- **Test Steps:**

  1.Open the Gmail login page and enter valid credentials. Access the inbox.

  2.Send an email to the Gmail address from another email account.

  3.Return to the Gmail inbox and check for the received email.
- **Expected Result:** The email sent from the alternate account should appear in the Gmail inbox without delays or issues. This test verifies that Gmail correctly receives and displays incoming emails as expected.

2. **Inbox Organization and Display:**

- **Description:** To ensure that the Gmail inbox correctly organizes and displays received emails.
- **Test Steps:**
  1.Verify that recently received or unread emails are highlighted in the inbox.
  2.Confirm that all emails (both read and unread) are displayed in the inbox.

  3.Check that the sender's name, email subject, date, and time are in bold, while previews are not.

  4.Verify that attachment icons appear next to the preview text when attachments are present.

3. Navigation to Email Content:

- **Description:** To validate that clicking on a newly received email navigates the user to the email content.
- **Test Steps:**

  1.Open the Gmail inbox.

  2.Click on a newly received email.

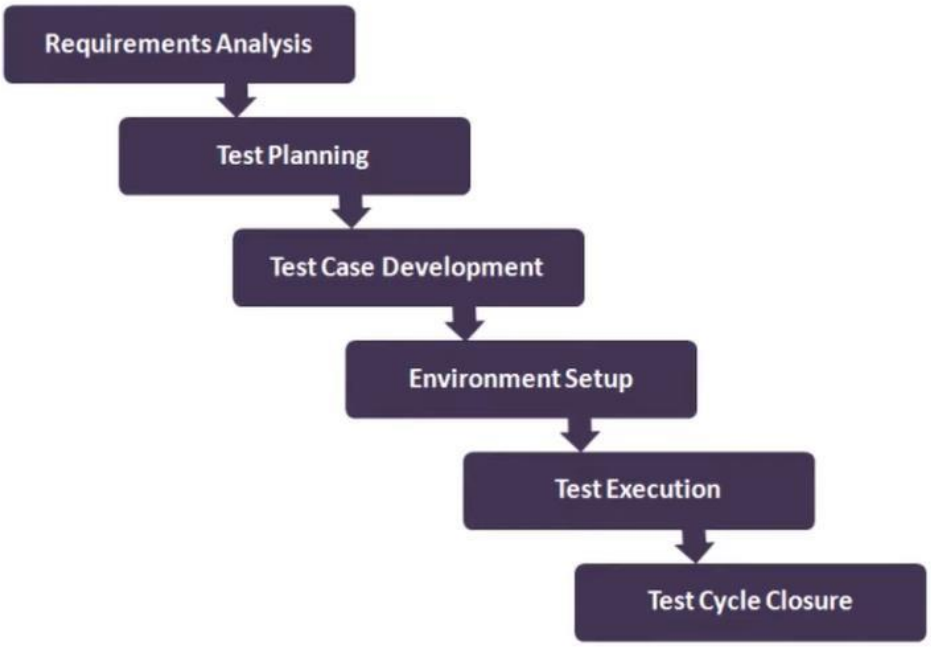
- **Expected Result:** The user should be seamlessly navigated to the content of the selected email

Experiment No - 4

Q. STLC - Software Testing Life Cycle.

Steps :

1. Requirement Analysis
2. Test Planning
3. Test Design
4. Test Environment Setup
5. Test Execution
6. Test Closure



1. Requirement Analysis:

In this phase, the requirements documents are analyzed and validated, and the scope of testing is defined.

2. Test Planning:

In this phase, **test plan strategy** is defined, estimation of **test effort** is defined along with automation strategy and tool selection is done.

3. Test Design:

In this phase test cases are designed; test data is prepared, and automation scripts are implemented.

4. Test Environment Setup:

A test environment closely simulating the real-world environment is prepared.

5. Test Execution:

To perform **actual testing** as per the test steps.

6. Test Closure:

Test Closure is final stage of STLC where we will make all details documentations which are required submit to client at time of software delivery. Such as test report, defect report, test cases summary, RTM details, release note.

SNO	PHASE	Input	Activities	Responsibility	Out Come
1	**Test Planning**	Project Plan	➢Identify the Resources	Test Lead/Team Lead (70%)	Test Plan Document
	What to test	Functional Requirements	➢Team Formation	Test Manager (30%)	
	How to test		➢Test Estimation		
	when to test		➢Preparation of Test Plan		
			➢Reviews on Test Plan		
			➢Test Plan Sign-off		
2	**Test Designing**	Project Plan	➢Preparation of Test Scenarios	Test Lead/Team Lead(30%)	Test Cases Document
		Functional Requirements	➢Preparation of Test Cases	Test Engineers(70%)	Traceability Matrix
		Test Plan	➢Reviews on Test Cases		
		Design Docs	➢Traceability Matrix		
		Use cases	➢Test Cases Sign-off		
3	**Test Execution**	Functional Requirements	➢Executing Test cases	Test Lead/Team Lead(10%)	Status/Test Reports
		Test Plan	➢Preparation of Test Report/Test Log	Test Engineers (90%)	
		Test Cases	➢Identifying Defects		
		Build from Development Team			
4	**Defect Reporting & Tracking**	Test Cases	➢Preparation of Defect Report	Test Lead/Team Lead(10%)	Defect Report
		Test Reports/Test Log	➢Reporting Defects to Developers	Test Engineers (90%)	
5	**Test Closure/Sign-Off**	Test Reports	➢Analyzing Test Reports	Test Lead/Test Manger(70%)	**Test Summary Reports**
		Defect Reports	➢Analyzing Bug Reporting	Test Enginners(30%)	
			➢Evaluating Exit Criteria		

Experiment No - 5

Q. Write Test Cases for "Contact Us" page :

It's important to consider various aspects, including functionality, usability, and design. Here are some test cases grouped by category:

1. Functional Test Cases

Test Case 1: Form Submission

Description : Verify that the form can be submitted successfully.

Steps : Fill in all fields with valid data and click the "Submit" button.

Expected Result : A confirmation message appears, and the user is redirected to a thank-you page.

Test Case 2 : Required Fields Validation

Description : Check that the required fields cannot be left blank.

Steps : Leave required fields empty and click "Submit."

Expected Result : An error message appears next to each required field.

Test Case 3 : Email Format Validation

Description : Ensure that the email field accepts valid email formats.

Steps : Enter an invalid email format (e.g., "test@com") and click "Submit."

Expected Result : An error message prompts the user to enter a valid email address.

Test Case 4 : Maximum Character Limit

Description : Verify that the message field has a character limit.

Steps : Enter a message that exceeds the maximum character limit and click "Submit."

Expected Result : An error message indicates that the message is too long.

2. Usability Test Cases

Test Case 6 : Accessibility Compliance

Description : Check if the form is accessible for users with disabilities.

Steps : Use screen reader software to navigate the page.

Expected Result : All elements are properly announced, and the form is navigable.

Test Case 7 : Placeholder Text

Description : Verify that placeholder text provides guidance for each input field.

Steps : Review the placeholder text in each field.

Expected Result : Placeholder text clearly indicates what information is needed.

Test Case 8 : Field Labels

Description : Ensure that all input fields have clear labels.

Steps : Check each field for associated labels.

Expected Result : Each input field has a visible label that describes its purpose.

3. Design Test Cases

Test Case 9 : Responsive Design

Description : Verify that the contact form is responsive across different devices.

Steps : Access the form on mobile, tablet, and desktop devices.

Expected Result : The layout adjusts appropriately without losing functionality.

Test Case 10 : Visual Design Consistency

Description : Check for consistency in design elements (fonts, colors, buttons).

Steps : Compare the contact form design with the overall website design.

Expected Result : The contact form matches the overall website style guide.

Test Case 11 : Error Message Display

Description : Ensure that error messages are visually distinct and easy to understand.

Steps : Trigger validation errors by submitting the form with invalid data.

Expected Result : Error messages appear in a contrasting color and are positioned near the corresponding fields.

4. Security Test Cases

Test Case 12: SQL Injection

Description : Test the form against SQL injection attacks.

Steps : Input SQL code (e.g., "'); DROP TABLE users; --") in the fields.

Expected Result : The system does not execute the SQL command and displays an error.

Test Case 13 : Cross-Site Scripting (XSS)

Description : Check if the form sanitizes input to prevent XSS attacks.

Steps : Enter a script tag (e.g., `<script>alert('XSS');</script>`) in the message field.

Expected Result : The system neutralizes the script and does not execute it.

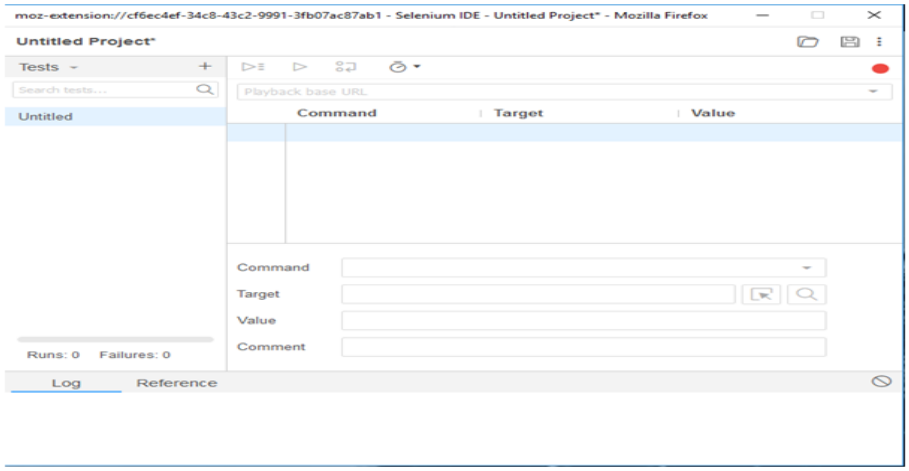
These test cases provide a comprehensive framework for evaluating a "Contact Us" page from multiple perspectives.

Experiment No - 6

Q. Selenium Integrated Development Environment (IDE)

Selenium IDE (Integrated Development Environment) is an open source web automation testing tool under the Selenium Suite. Unlike Selenium WebDriver and RC, it does not require any programming logic to write its test scripts rather you can simply record your interactions with the browser to create test cases. Subsequently, you can use the playback option to re-run the test cases.

The following image shows the default interface of Selenium IDE:



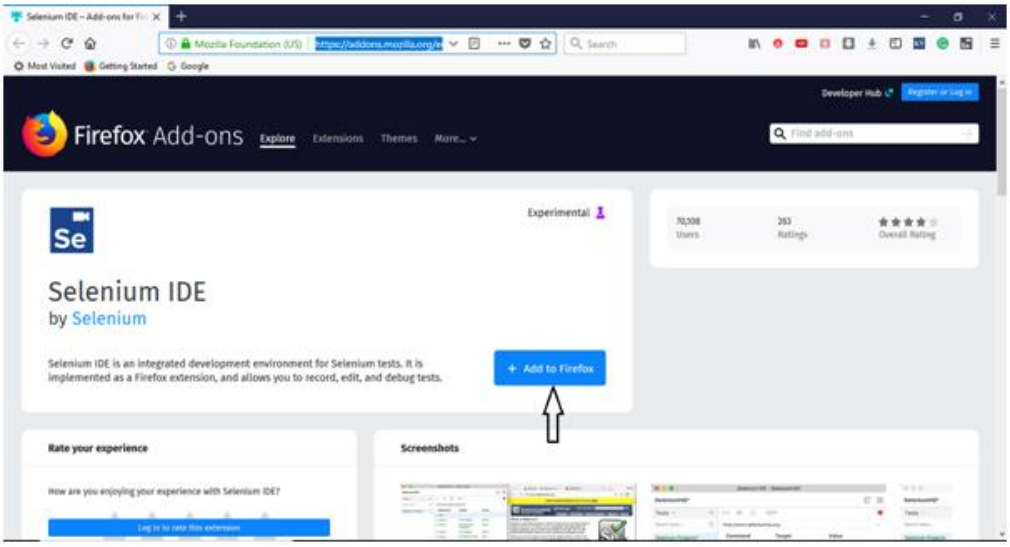
Selenium IDE-Installation

Since, Selenium IDE is available only as Firefox and Chrome plug-in, we assume that you have already installed Mozilla Firefox browser in your system. However, you can download the latest version of Firefox through their official website provided under the link given below.

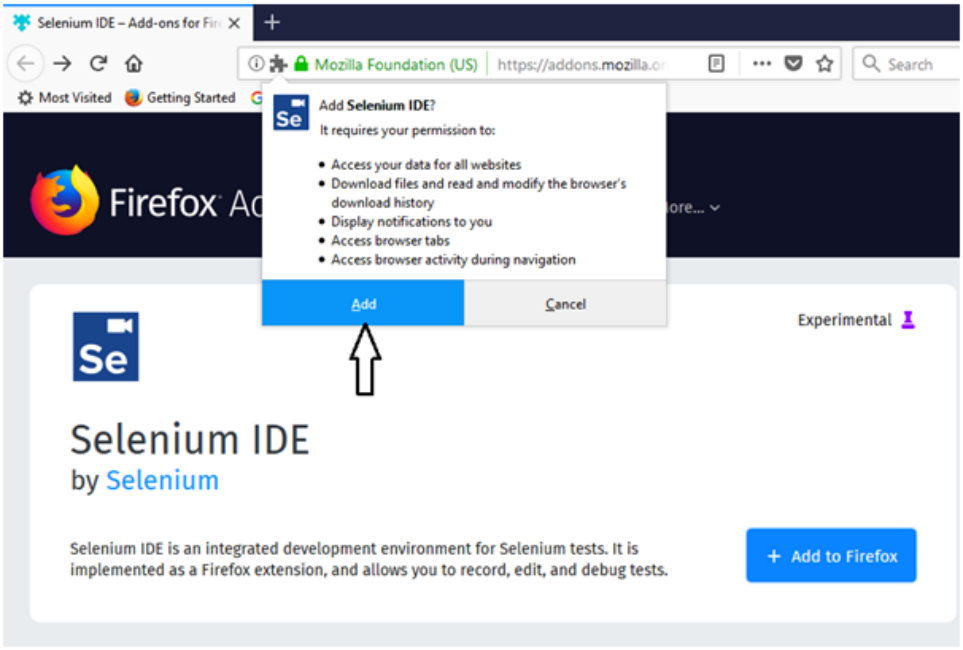
https://www.mozilla.org/en-US/firefox/new/

Selenium IDE Download and Install

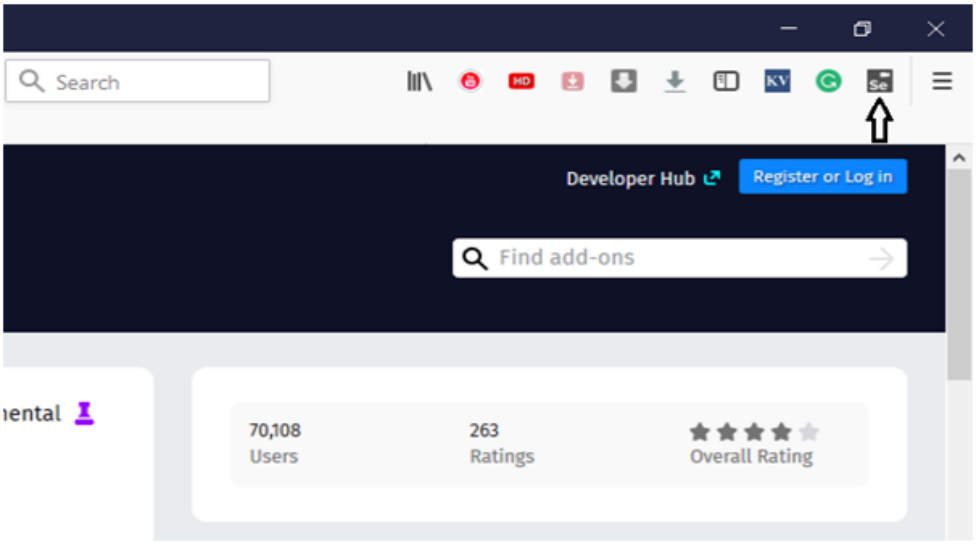
- Launch Mozilla Firefox browser.
- Open URLhttps://addons.mozilla.org/en-us/firefox/addon/selenium-ide/It will redirect you to the official add-on page of Firefox.
- Click on "Add to Firefox" button.



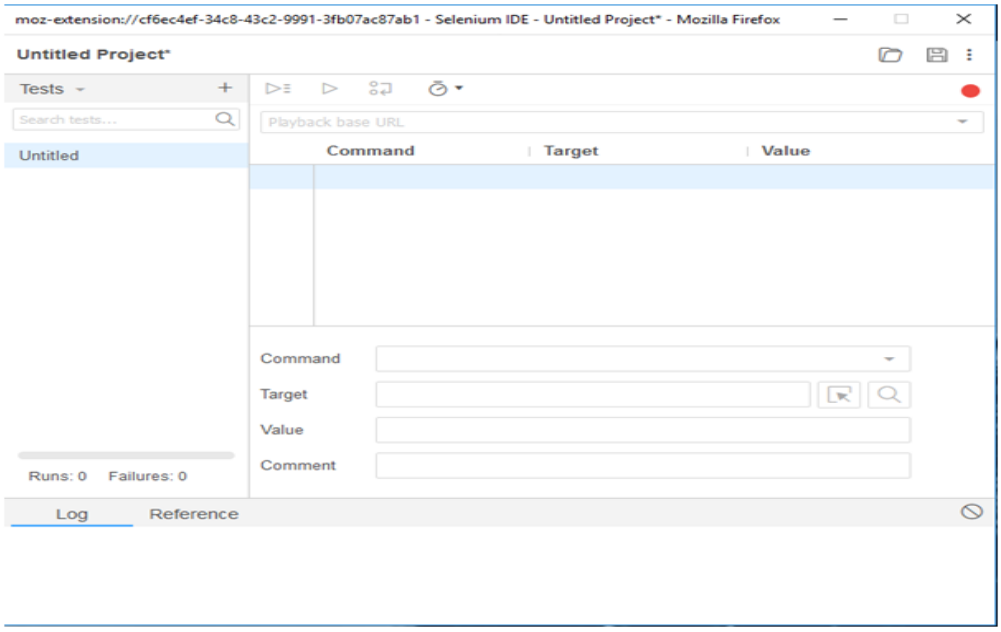
- A pop-up dialog box will be appeared asking you to add Selenium IDE as extension to your Firefox browser.
- Click on "Add" button.



- Restart you Firefox browser.
- Go to the top right corner on your Firefox browser and look for the Selenium IDE icon.



- Click on that icon to launch Selenium IDE.



Experiment No - 7

Q. White box testing and Black box testing.

White Box testing

The term 'white box' is used because of the internal perspective of the system. The **clear box or white box, or transparent box** name denotes the ability to see through the software's outer shell into its inner workings.

It is performed by Developers, and then the software will be sent to the testing team, where they perform black-box testing. The main objective of white-box testing is to test the application's infrastructure. It is done at lower levels, as it includes unit testing and integration testing. It requires programming knowledge, as it majorly focuses on code structure, paths, conditions, and branches of a program or software. The primary goal of white-box testing is to focus on the flow of inputs and outputs through the software and strengthening the security of the software.

It is also known as structural testing, clear box testing, code-based testing, and transparent testing. It is well suitable and recommended for algorithm testing.

Black Box testing

The primary source of black-box testing is a specification of requirements that are stated by the customer. It is another type of manual testing. It is a software testing technique that examines the functionality of the software without knowing its internal structure or coding. It does not require programming knowledge of the software. All test cases are designed by considering the input and output of a particular function. In this testing, the test engineer analyzes the software against requirements, identifies the defects or bugs, and sends it back to the development team.

In this method, the tester selects a function and gives input value to examine its functionality, and checks whether the function is giving the expected output or not. If the function produces the correct output, then it is passed in testing, otherwise failed.

Black box testing is less exhaustive than White Box and Grey Box testing methods. It is the least time-consuming process among all the testing processes. The main objective of implementing black box testing is to specify the business needs or the customer's requirements.

In other words, we can say that black box testing is a process of checking the functionality of an application as per the customer's requirement. Mainly, there are three types of black-box testing**: functional testing, Non-Functional testing,** and **Regression testing**. Its main objective is to specify the business needs or the customer's requirements.

Experiment No - 8

Q. Software Testing Terminologies:

A. Regression Testing

B. Re-testing

A. Regression testing :

Testing conducted on modified build (updated build) to make sure there will not be an impact on existing functionality because of changes like adding/deleting/modifying features. Also, we can say Smoke Testing is a small part of regression testing.

1. Unit regression testing type:

i. Testing only the changes/modification done by the developer.

2. Regional Regression testing type:

i. Testing the modified module along with the impacted modules.

ii. Impact Analysis meeting conducts to identify impacted modules with QA and developer.

3. Full Regression type:

i. Testing the main feature and remaining part of the application.

ii. Example: The developer has done changes in many modules, instead of identifying
impacted modules, we perform one round of full regression.

❖ **When we performed regression testing:**

- After Defect get Fixed and Retesting is done.
- Modification, Updating, change request into Existing functionality.
- After addition of new functionality.
- Before release.
- After release.
- Data Migration.
- After change in Environment

B. Re-Testing:

1. Whenever the developer fixed a bug, the tester will test the bug fix called re-testing.
2. Tester closes the bug if worked otherwise re-open and send to a developer.
3. To ensure that the defects which were found and posted in the earlier build were fixed or not in the current build.
4. Example:

i. Build 1.0 was released, test team found some defects (Defect ID 1.0.1, 1.0.2) and posted them.

ii. Build 1.1 was released, now testing the defects 1.0.1 and 1.0.2 in this build is retesting.

Re-Testing VS. Regression Testing: