**REGISTRATION : SELINIUM AUTOMATED TEST CASE**

**LOGIN : SELENIUM AUTOMATED TEST CASE**

*Software Testing Methodology Group Project Report Submitted by*

**Mohammad Rizwanullah (208W1A1299)**

**Motamarri Jaya Naga Venkata Sai (208W1A12A0)**

****

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**V R SIDDHARTHA ENGINEERING COLLEGE**

**(AUTONOMOUS - AFFILIATED TO JNTU-K, KAKINADA)**

**Approved by AICTE &Accreted by NBA**

**KANURU, VIJAYAWADA-7**

**ACADEMIC YEAR**

**(2023 - 24)**

**TABLE OF CONTENTS**

**1. INSTALLATION OF SELENIUM**

**2. PROBLEM STATEMENT**

**3. SCOPE OF THE PROJECT**

**4. SOFTWARE REQUIREMENTS**

**5. IMPORTANTANCE OF SELENIUM**

**6. VARIOUS TYPES OF LOCATORS**

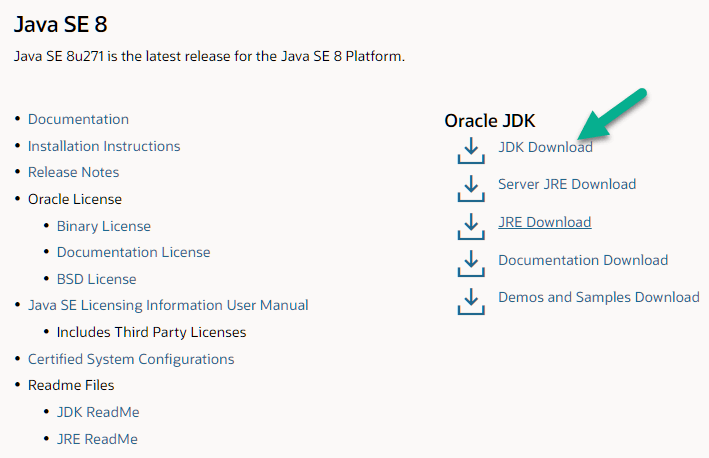
**7. IMPLEMENTATION STEPS**

**8. RESULT ANALYSIS**

**TASK – 1 : REGISTRATION AUTOMATED SELENIUM TEST CASES**

**SELENIUM INSTALLATION**

1. First You have install Java Version 8 or above
2. Go to link <https://www.oracle.com/java/technologies/downloads/> . Click on JDK Download for Java download JDK 8.



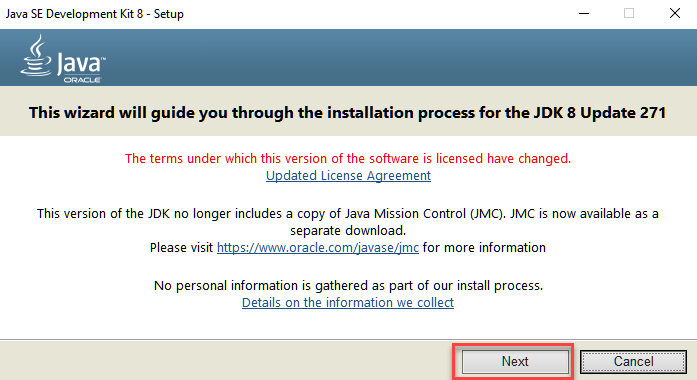
1. Next, Accept License Agreement
2. Download Java 8 JDK for your version 32 bit or JDK download 64 bit.



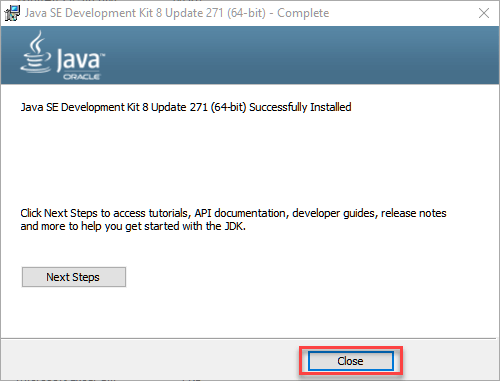
1. When you click on the Installation link the popup will be open. Click on I reviewed and accept the Oracle Technology Network License Agreement for Oracle Java SE development kit and you will be redirected to the login page. If you don’t have an oracle account you can easily sign up by adding basics details of yours.



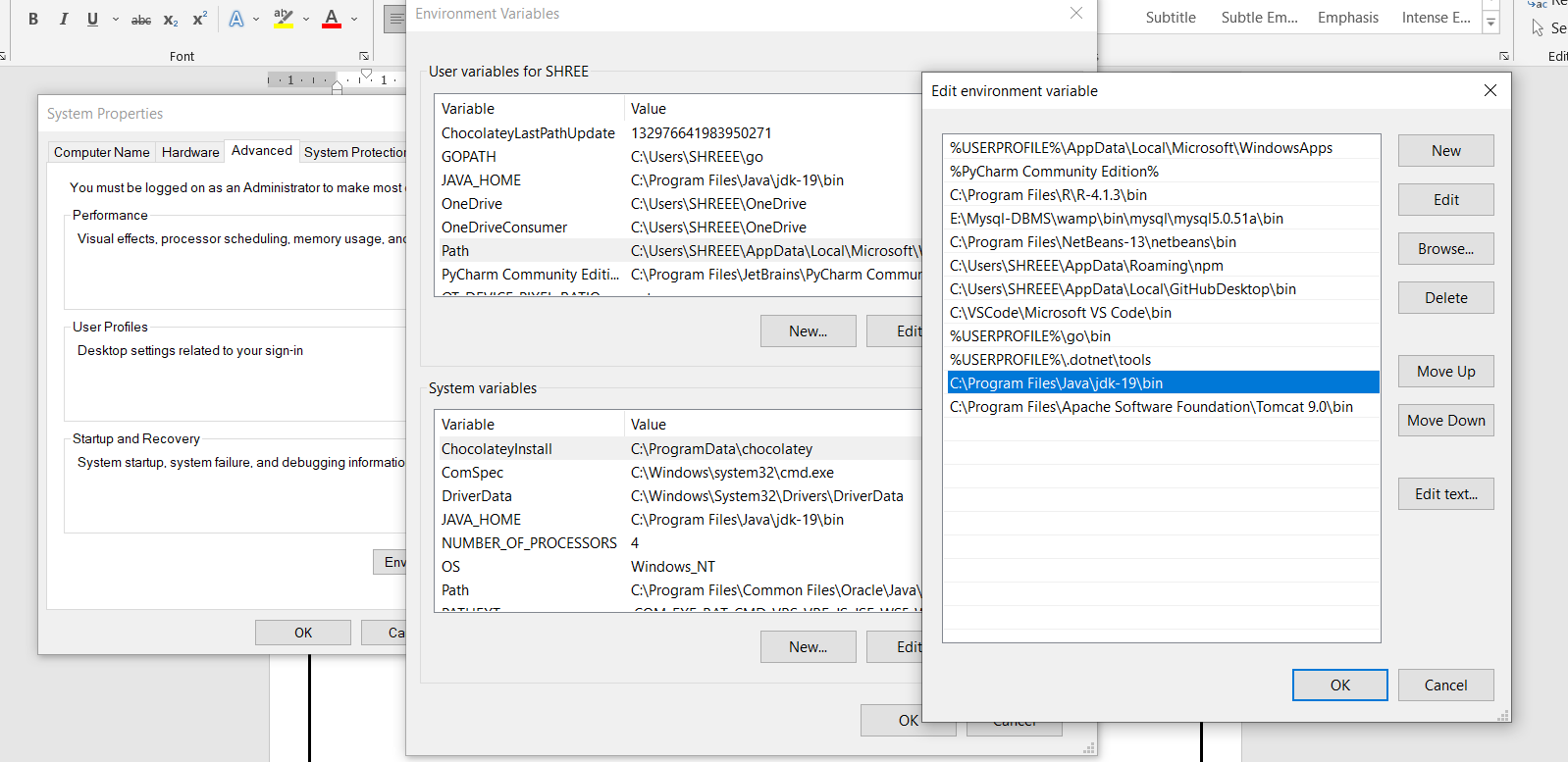
1. You will be required to create an Oracle Account to start Java 8 download of the file.
2. Once the Java JDK 8 download is complete, run the exe for install JDK. Click Next



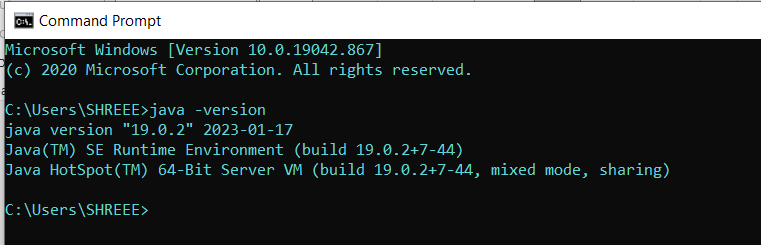
1. Once you install Java in windows, click Close



1. Click on Windows Button and Type “Environmental variables “ and then press enter
2. And again click on “environmental variables” button and then add an path of the your jdk version .
3. Add your jdk path to the user varibles section .
4. Your jdk path will be like this “C://program files/java/jdk version/bin/ “ in windows operating system.

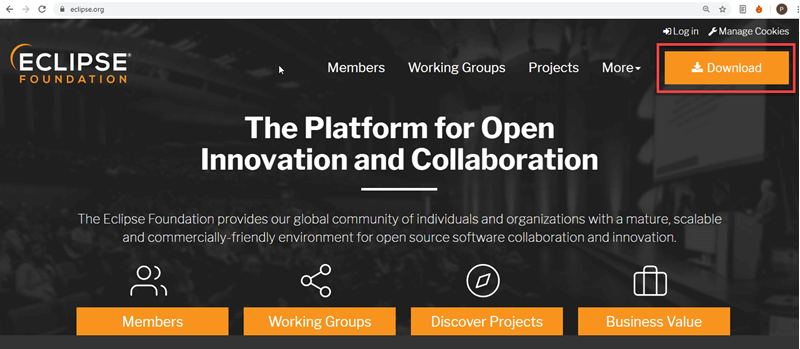


1. Now check in windows command prompt and type “java -version” you get your jdk and jre versions.

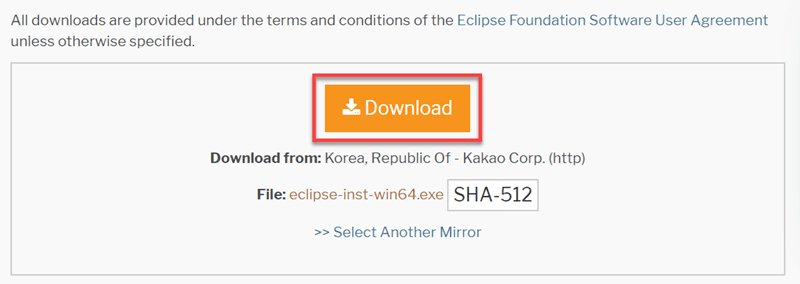


**Now, Eclipse Installation :**

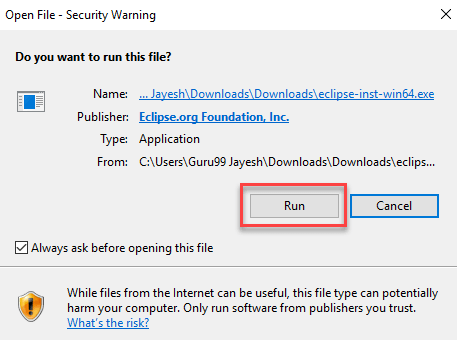
1. Installing Eclipse
2. Open your browser and type https://www.eclipse.org/
3. And then click on download button.



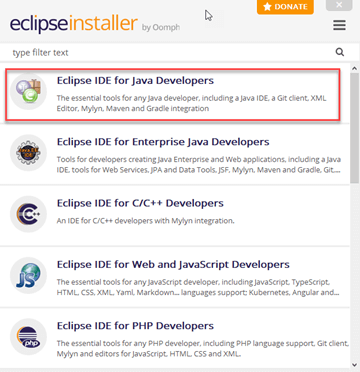
1. Based on your click on 32-bit version or 64-bit version.



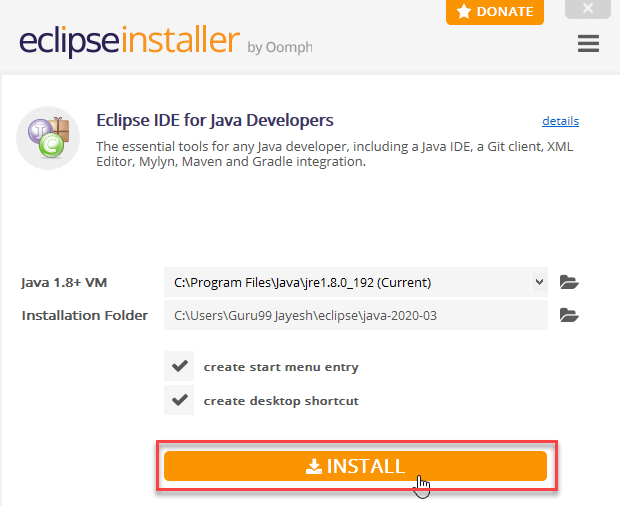
1. Now in downloads folder double click on that .exe file to run it.
2. And then click on run button.



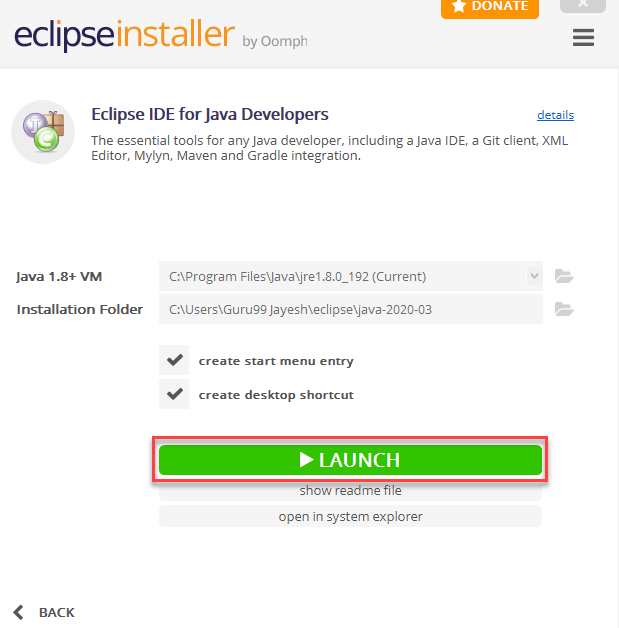
1. Click on “eclipse ide for java developers”

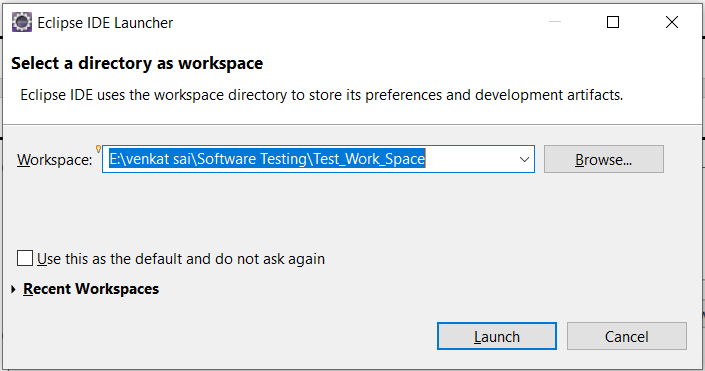


1. Click on “INSTALL” button

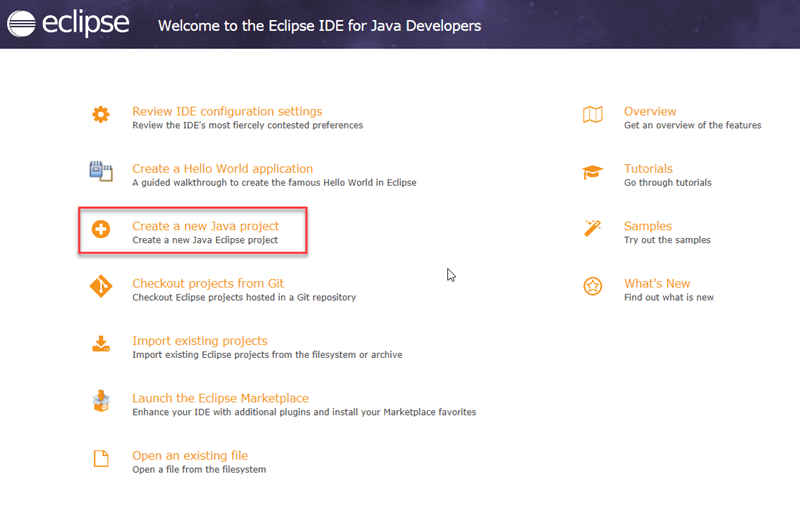


1. Click on “Launch Button”





1. The Final Output after the installation will be like below picture.



**Problem Statement – 1**

**5 .** Write the Automation Test Scripting for the following Scenarios.

a) Verify if Special Characters are allowing in the First and Last name

b) Verify if Email field is accepting only numbers as Email (without @ and

.com or .co.in etc.,)

c) Verify if Email field is accepting only Special Characters as Email (without

@and .com or .co.in etc.,)

**Scope Of The Project**

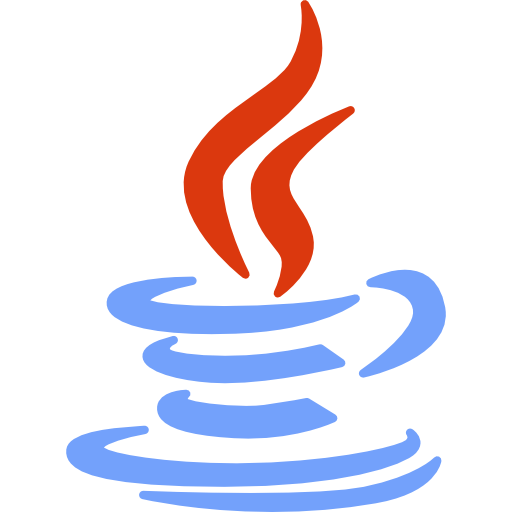
1. The project involves testing the input validation functionality of a web application. Specifically, the following scenarios need to be tested:
2. Verify if Special Characters are allowing in the First and Last name: This test case involves checking if special characters such as !@#$%^&\*()\_+ are allowed in the First and Last name fields. The expected behavior is that special characters should not be allowed, and an appropriate error message should be displayed to the user.
3. Verify if Email field is accepting only numbers as Email (without @ and .com or .co.in etc.,): This test case involves testing the email validation functionality by entering only numbers in the email field. The expected behavior is that an error message should be displayed to the user, indicating that a valid email address is required.
4. Verify if Email field is accepting only Special Characters as Email (without @and .com or .co.in etc.,): This test case involves testing the email validation functionality by entering special characters such as !@#$%^&\*()\_+ in the email field. The expected behavior is that an error message should be displayed to the user, indicating that a valid email address is required.

**SOFTWARE REQUIREMENTS**

1. Eclipse Integrated Developed Environment with version 2022 – 2023



1. Selenium, WebDriver Manger, apache, testing-metrics
2. Install Testng Package from the Eclipse market Place
3. Java Development kit of version 8
4. Java runtime environment of version 19 (minimum requirement )



1. System with minimum 4GB Ram



**SELENIUM IMPORTANCE**

1. Software testing is an essential aspect of software development that ensures the quality and reliability of the product. In recent years, automated testing has gained immense popularity due to its ability to increase testing efficiency, reduce costs, and minimize errors. One of the most popular automated testing tools in the industry is Selenium. Selenium is an open-source tool that automates web browsers and can be used for testing web applications. In this essay, we will discuss the importance of automated Selenium testing in software development.
2. One of the most significant advantages of using Selenium for testing is its ability to reduce testing time. Automated testing is much faster than manual testing because it can execute test cases much more quickly and accurately. Automated tests can also run simultaneously on multiple machines, which saves a lot of time and effort.
3. Another significant advantage of Selenium testing is its ability to increase testing coverage. Automated tests can cover more scenarios and edge cases than manual tests, which is impossible to achieve manually. Selenium testing can also be used for regression testing, which ensures that new code changes do not affect the existing functionality of the application.
4. Selenium testing also reduces the risk of human errors in testing. Manual testing is prone to errors due to the repetitive nature of the task, and testers can easily miss critical defects. Automated testing with Selenium eliminates the possibility of human errors and ensures that all test cases are executed consistently.
5. Automated Selenium testing also helps in cost reduction. Automated tests can be reused multiple times, which saves a lot of time and effort in executing the same tests manually. Selenium testing also reduces the need for a large number of testers, which results in significant cost savings.
6. Selenium testing also provides a better ROI (Return on Investment) for software development. The increased testing efficiency and coverage achieved through Selenium testing result in better quality software products. This leads to improved customer satisfaction and fewer customer complaints, which ultimately results in increased revenue for the company.

**VARIOUS TYPES OF LOCATORS**

In Selenium testing, locators are used to identify web elements on a webpage. Locators are essential to interact with web elements such as buttons, links, input fields, etc. There are various types of locators available in Selenium, such as ID, name, class name, tag name, link text, and partial link text. In this essay, we will discuss class locators in Selenium testing.

**CLASS LOCATOR :**

1. Class locators are used to locate web elements using the class attribute of the HTML tag. The class attribute is used to define a class for an HTML element, which can be used to apply styling and functionality to that element. Class locators are very useful when multiple elements have the same tag name, and you want to locate a specific element based on its class.
2. To use class locators in Selenium, you need to use the driver.findElement() method, which takes a By object as a parameter. The By object is used to specify the type of locator you want to use. To use the class locator, you need to create a new By object using the By.className() method, which takes the name of the class as a parameter.
3. For example, if you want to locate a button on a webpage with a class name of "btn-primary," you can use the following code:

|  |
| --- |
| WebElement button = driver.findElement(By.className("btn-primary")); |

1. The above code will locate the button element with the class name "btn-primary" on the webpage and assign it to the WebElement object named "button."
2. One thing to keep in mind while using class locators is that the class name must be unique on the webpage. If there are multiple elements with the same class name, the class locator will locate the first element that matches the class name.

**ID LOCATOR :**

1. The ID locator is one of the most popular and frequently used locators in Selenium testing. It is used to locate a web element based on its unique identifier attribute, which is commonly referred to as the "ID." The ID attribute is a unique identifier assigned to a web element by the developer, which can be used to identify that element uniquely.
2. To use the ID locator in Selenium, you need to use the driver.findElement() method, which takes a By object as a parameter. The By object is used to specify the type of locator you want to use. To use the ID locator, you need to create a new By object using the By.id() method, which takes the ID of the web element as a parameter.

For example, if you want to locate a button on a webpage with an ID of "submit-

btn," you can use the following code :

|  |
| --- |
| WebElement button = driver.findElement(By.id("submit-btn")); |

1. The above code will locate the button element with the ID "submit-btn" on the webpage and assign it to the WebElement object named "button."
2. One thing to keep in mind while using the ID locator is that the ID attribute must be unique on the webpage. If there are multiple elements with the same ID, the ID locator will locate the first element that matches the ID.

**XPATH LOCATOR :**

1. XPath is a language used to navigate and select elements in an XML document. In Selenium, XPath is used to locate web elements on a webpage by searching for elements using their attributes, such as ID, name, class, etc. XPath can be an absolute path or a relative path, depending on whether it starts with a single slash (/) or a double slash (//).
2. To use the XPath locator in Selenium, you need to use the driver.findElement() method, which takes a By object as a parameter. The By object is used to specify the type of locator you want to use. To use the XPath locator, you need to create a new By object using the By.xpath() method, which takes the XPath expression as a parameter. For example, if you want to locate a button on a webpage with the text "Submit," you can use the following code :

|  |
| --- |
| WebElement button = driver.findElement(By.xpath("//button[text()='Submit']")); |

1. Syntax : //tagname[@attributename = 'value']
2. Multiple Attributes : //input[@type='text'][@aria-label='First name']

**Xpath Axes :**

1) parent Node ====> Child Node

2) //input[@type='checkbox']following::label

3) //p[text()='vrsec'] ==> for paragraphs

4) //tagname[contains@atrribute='value']

5) //p[contains(text(), 'vrsec']

1. XPath locators provide a lot of flexibility and power in locating web elements on a webpage. You can use XPath to locate elements based on their attributes, position, and even their relationships with other elements on the webpage. However, XPath locators can be slower than other types of locators, especially when using complex expressions or navigating large DOM trees.

**CSS LOCATORS :**

1. CSS (Cascading Style Sheets) is a style sheet language used to describe the presentation of a webpage. In Selenium testing, CSS locators are used to identify web elements on a webpage by searching for elements using their CSS attributes, such as ID, class, tag name, etc.
2. CSS locators can be used in two ways: direct and indirect. Direct CSS locators select elements based on their attributes, while indirect CSS locators select elements based on their relationships with other elements on the webpage.
3. To use CSS locators in Selenium, you need to use the driver.findElement() method, which takes a By object as a parameter. To use the CSS locator, you need to create a new By object using the By.cssSelector() method, which takes the CSS expression as a parameter. For example, if you want to locate a button on a webpage with the class "submit-btn," you can use the following code :

|  |
| --- |
| WebElement button = driver.findElement(By.cssSelector(".submit-btn")); |

1. The above code will locate the button element with the class "submit-btn" on the webpage and assign it to the WebElement object named "button."
2. CSS locators provide a lot of flexibility and power in locating web elements on a webpage. You can use CSS to locate elements based on their attributes, position, and even their relationships with other elements on the webpage. However, CSS locators can be slower than other types of locators, especially when using complex expressions or navigating large DOM trees.

**CSS Selector Syntax :**

1) tagname[attributename='value'] ===> input[type='text']

2) tagname#idname :===> id selectors

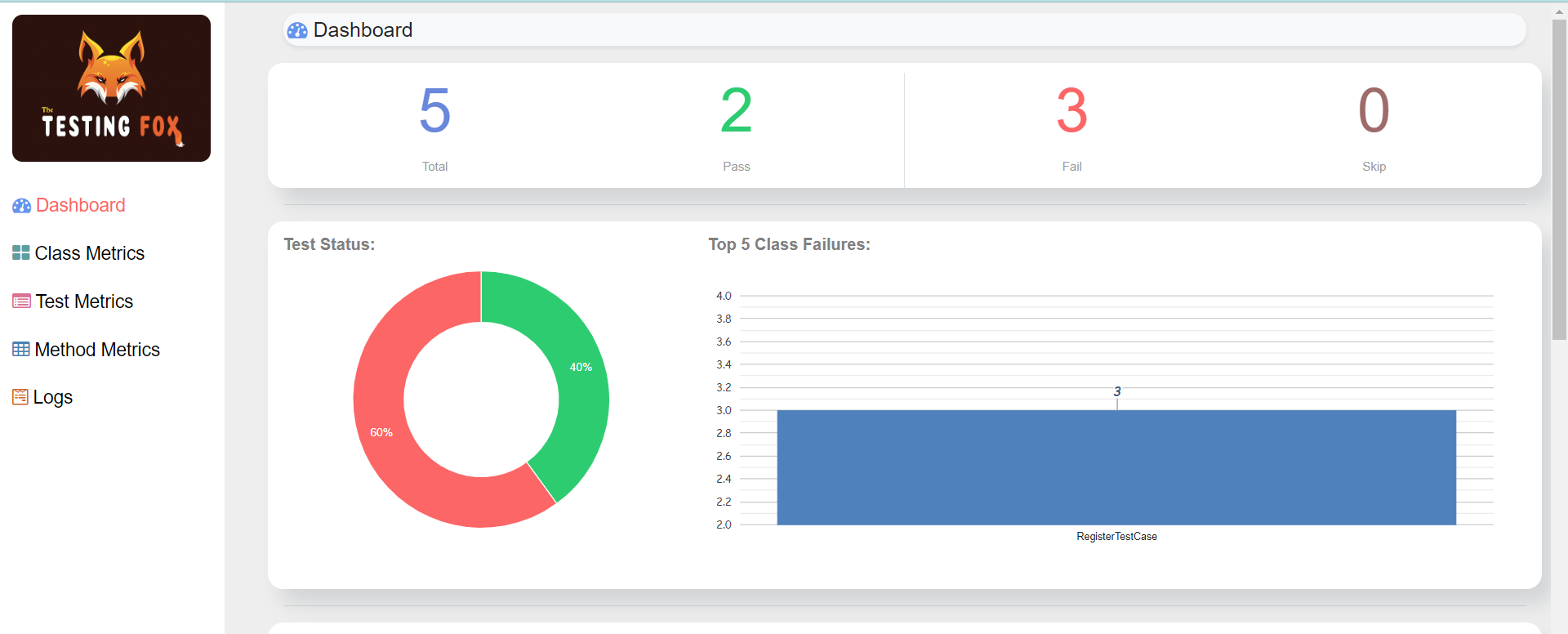
3) startswith (^) , Endswith ($), contains (\*)

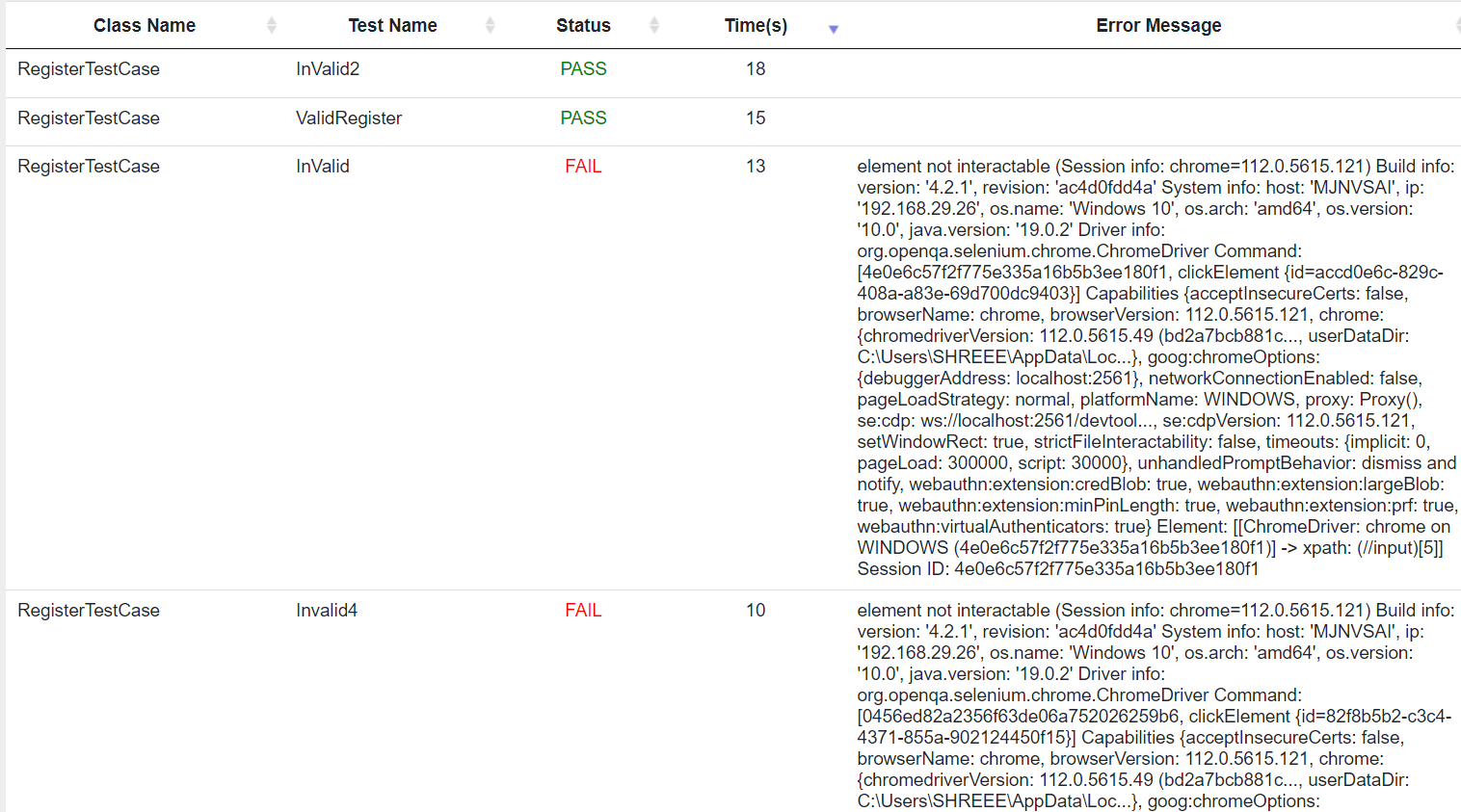
4) input[id^='vrsec'], input[id$='vrsec'], input[id\*='vrsec']

**IMPLEMENTATION OF CODE**

|  |
| --- |
| REGISTER TEST CASES  package loginCase;  import org.testng.annotations.Test;  import org.testng.annotations.Test;  import java.io.File;  import java.io.IOException;  import java.lang.\*;  import org.apache.commons.io.FileUtils;  import org.openqa.selenium.By;  import org.openqa.selenium.OutputType;  import org.openqa.selenium.TakesScreenshot;  import org.openqa.selenium.WebDriver;  import org.openqa.selenium.chrome.ChromeDriver;  import org.openqa.selenium.chrome.ChromeOptions;  import org.openqa.selenium.firefox.FirefoxDriver;  import org.openqa.selenium.firefox.FirefoxOptions;  import org.testng.Assert;  import org.testng.annotations.AfterTest;  import org.testng.annotations.BeforeTest;  import org.testng.annotations.Test;  import io.github.bonigarcia.wdm.WebDriverManager;  public class LoginTestCase  {  WebDriver driver;  @BeforeTest  public void beforeTest()  {  WebDriverManager.firefoxdriver().setup();  FirefoxOptions options = new FirefoxOptions();  driver = new FirefoxDriver();  }  @Test(description = "Valid Username and Valid Password", priority = 6)  public void ValidLogin() throws Exception  {  driver.get("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login");  driver.manage().window().maximize();  Thread.sleep(5000);  String usernamelog = "Admin";  String passwordlog = "admin123";  System.out.println("Test Case 1");  driver.findElement(By.name("username")).sendKeys(usernamelog);  driver.findElement(By.name("password")).sendKeys(passwordlog);  Thread.sleep(5000);  driver.findElement(By.xpath("//button[@type='submit']")).click();  Thread.sleep(5000);  String actual\_message= driver.findElement(By.xpath("//\*[@id=\"app\"]/div[1]/div[1]/header/div[1]/div[1]/span/h6")).getText();  String expected\_message="Dashboard";  System.out.println(actual\_message);  Assert.assertEquals(actual\_message, expected\_message);  afterTest();  }  @Test(description = "Valid Username and In\_Valid Password", priority = 2)  public void Invalid1() throws Exception  {  beforeTest();  driver.get("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login");  driver.manage().window().maximize();  Thread.sleep(5000);  String usernamelog = "Admin";  String passwordlog = "Admin123";  System.out.println("Test Case 2");  driver.findElement(By.name("username")).sendKeys(usernamelog);  driver.findElement(By.name("password")).sendKeys(passwordlog);  Thread.sleep(5000);  driver.findElement(By.xpath("//button[@type='submit']")).click();  Thread.sleep(5000);  String actual\_message= driver.findElement(By.xpath("//\*[@id=\"app\"]/div[1]/div[1]/header/div[1]/div[1]/span/h6")).getText();  String expected\_message="Dashboard";  System.out.println(actual\_message);  Assert.assertEquals(actual\_message, expected\_message);  afterTest();  }  @Test(description = "In\_Valid Username and Valid Password", priority = 3)  public void Invalid2() throws Exception  {  beforeTest();  driver.get("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login");  driver.manage().window().maximize();  Thread.sleep(5000);  String usernamelog = "Admin\_sai";  String passwordlog = "admin123";  System.out.println("Test Case 3");  driver.findElement(By.name("username")).sendKeys(usernamelog);  driver.findElement(By.name("password")).sendKeys(passwordlog);  Thread.sleep(5000);  driver.findElement(By.xpath("//button[@type='submit']")).click();  Thread.sleep(5000);  String actual\_message= driver.findElement(By.xpath("//\*[@id=\"app\"]/div[1]/div[1]/header/div[1]/div[1]/span/h6")).getText();  String expected\_message="Dashboard";  System.out.println(actual\_message);  Assert.assertEquals(actual\_message, expected\_message);  afterTest();  }  @Test(description = "In\_Valid Username and In\_Valid Password", priority = 4)  public void Invalid3() throws Exception  {  beforeTest();  driver.get("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login");  driver.manage().window().maximize();  Thread.sleep(5000);  String usernamelog = "Admin\_sai";  String passwordlog = "admin\_sai";  System.out.println("Test Case 4");  driver.findElement(By.name("username")).sendKeys(usernamelog);  driver.findElement(By.name("password")).sendKeys(passwordlog);  Thread.sleep(5000);  driver.findElement(By.xpath("//button[@type='submit']")).click();  Thread.sleep(5000);  String actual\_message= driver.findElement(By.xpath("//\*[@id=\"app\"]/div[1]/div[1]/header/div[1]/div[1]/span/h6")).getText();  String expected\_message="Dashboard";  System.out.println(actual\_message);  Assert.assertEquals(actual\_message, expected\_message);  afterTest();  }  @Test(description = "Empty Username and Empty Password", priority = 5)  public void Invalid4() throws Exception  {  beforeTest();  driver.get("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login");  driver.manage().window().maximize();  Thread.sleep(5000);  String usernamelog = "";  String passwordlog = "";  System.out.println("Test Case 5");  driver.findElement(By.name("username")).sendKeys(usernamelog);  driver.findElement(By.name("password")).sendKeys(passwordlog);  Thread.sleep(5000);  driver.findElement(By.xpath("//button[@type='submit']")).click();  Thread.sleep(5000);  String actual\_message= driver.findElement(By.xpath("//\*[@id=\"app\"]/div[1]/div[1]/header/div[1]/div[1]/span/h6")).getText();  String expected\_message="Dashboard";  System.out.println(actual\_message);  Assert.assertEquals(actual\_message, expected\_message);  }  @AfterTest()  public void afterTest(){  Thread.sleep(1000);  driver.quit(); } } |

**RESULT ANALYSIS**

****

****

**PROBLEM STATEMENT - 2**

Write the Automation Test Scripting for the following Scenarios :

a) Verify if a user will be able to login with a valid Email and valid

Password

b) Verify if a user cannot login with a Invalid Email and valid

Password.

c) Verify if a user cannot login with a valid Email and Invalid Password.

**SCOPE OF THE PROJECT**

1. The objective of the project is to test the login functionality of the target application.
2. The scope of the project is limited to testing the login page's behavior with valid and invalid input values.
3. The testing should cover three scenarios: successful login with valid credentials, unsuccessful login with an invalid email and valid password, and unsuccessful login with a valid email and invalid password.

**IMPLEMENTATION STEPS**

|  |
| --- |
| LOGIN TEST CASES  package RegisterCase;  import org.testng.annotations.Test;  import org.testng.annotations.Test;  import java.io.File;  import java.io.IOException;  import java.lang.\*;  import org.apache.commons.io.FileUtils;  import org.openqa.selenium.By;  import org.openqa.selenium.OutputType;  import org.openqa.selenium.TakesScreenshot;  import org.openqa.selenium.WebDriver;  import org.openqa.selenium.chrome.ChromeDriver;  import org.openqa.selenium.chrome.ChromeOptions;  import org.testng.Assert;  import org.testng.annotations.AfterTest;  import org.testng.annotations.BeforeTest;  import org.testng.annotations.Test;  import io.github.bonigarcia.wdm.WebDriverManager;  public class RegisterTestCase {  WebDriver driver;  @BeforeTest  public void beforeTest() {  WebDriverManager.chromedriver().setup();  ChromeOptions options = new ChromeOptions();  options.addArguments("--remote-allow-origins=\*","ignore-certificate-errors");  driver = new ChromeDriver(options);  }    @Test(description = "Giving All Valid Values To the Form", priority = 1)  public void ValidRegister() throws Exception  {  driver.get("https://demowebshop.tricentis.com/register");  driver.manage().window().maximize();  Thread.sleep(5000);  driver.findElement(By.id("gender-male")).click();  driver.findElement(By.name("FirstName")).sendKeys("Sai");  driver.findElement(By.name("LastName")).sendKeys("MJNV");  driver.findElement(By.name("Email")).sendKeys("mjnvsai45@gmail.com");  driver.findElement(By.name("Password")).sendKeys("sai12345");  driver.findElement(By.name("ConfirmPassword")).sendKeys("sai12345");  System.out.print("Test Case - 1");  driver.findElement(By.name("register-button")).click();  driver.findElement(By.xpath("(//input)[5]")).click();  String actual\_message= driver.findElement(By.className("topic-html-content-header")).getText();  String expected\_message="Welcome to our store";  System.out.println(actual\_message);  Assert.assertEquals(actual\_message, expected\_message);  System.out.print("Test Case 1 passed");  afterTest(); }  @Test(description = "Giving All In Valid Values To the Form", priority = 2)  public void InValid() throws Exception  {  beforeTest();  driver.get("https://demowebshop.tricentis.com/register");  driver.manage().window().maximize();  Thread.sleep(5000);  driver.findElement(By.id("gender-male")).click();  driver.findElement(By.name("FirstName")).sendKeys("fhjsdvjhfb");  driver.findElement(By.name("LastName")).sendKeys("fsdfj sjdg"); driver.findElement(By.name("Email")).sendKeys("mjnvsai45@gmail.com");  driver.findElement(By.name("Password")).sendKeys("ertyui");  driver.findElement(By.name("ConfirmPassword")).sendKeys("ertyui");  System.out.print("Test Case - 2");  driver.findElement(By.name("register-button")).click();  driver.findElement(By.xpath("(//input)[5]")).click();  String actual\_message= driver.findElement(By.className("topic-html-content-header")).getText();  String expected\_message="Welcome to our store";  System.out.println(actual\_message);  Assert.assertEquals(actual\_message, expected\_message);  System.out.print("Test Case 2 passed");  afterTest(); }  @Test(description = "Special Characters are allowing in the First and Last name", priority = 3)  public void InValid2() throws Exception {  beforeTest();  driver.get("https://demowebshop.tricentis.com/register");  driver.manage().window().maximize();  Thread.sleep(5000);  driver.findElement(By.id("gender-male")).click(); driver.findElement(By.name("FirstName")).sendKeys("fhjsdv@#$%\_-hu");  driver.findElement(By.name("LastName")).sendKeys("fsdfj %$!@ sai \_-");  driver.findElement(By.name("Email")).sendKeys("jdfjsbfb@jbsajyuio.euir");  driver.findElement(By.name("Password")).sendKeys("ertyui");  driver.findElement(By.name("ConfirmPassword")).sendKeys("ertyui");  System.out.print("Test Case - 3");  driver.findElement(By.name("register-button")).click();  driver.findElement(By.xpath("(//input)[5]")).click();  String actual\_message= driver.findElement(By.className("topic-html-content-header")).getText();  String expected\_message="Welcome to our store";  System.out.println(actual\_message);  Assert.assertEquals(actual\_message, expected\_message);  System.out.print("Test Case 3 passed");  afterTest(); }    @Test(description = "if Email field is accepting only numbers as Email (without @ and .com or .co.in etc.,)", priority = 4)  public void Invalid3() throws Exception {  beforeTest();  driver.get("https://demowebshop.tricentis.com/register");  driver.manage().window().maximize();  Thread.sleep(5000);  driver.findElement(By.id("gender-male")).click(); driver.findElement(By.name("FirstName")).sendKeys("fhasdbv@#$%\_-hu");  driver.findElement(By.name("LastName")).sendKeys("fsdaabfj ^poi&%$!@ sai \_-");  driver.findElement(By.name("Email")).sendKeys("12345987");  driver.findElement(By.name("Password")).sendKeys("ertyui");  driver.findElement(By.name("ConfirmPassword")).sendKeys("ertyui");  System.out.print("Test Case - 4");  driver.findElement(By.name("register-button")).click();  driver.findElement(By.xpath("(//input)[5]")).click();  String actual\_message= driver.findElement(By.className("topic-html-content-header")).getText();  String expected\_message="Welcome to our store";  System.out.println(actual\_message);  Assert.assertEquals(actual\_message, expected\_message);    System.out.print("Test Case 4 passed");  afterTest(); }  @Test(description = "if Email field is accepting only Special Characters as Email (without @and .comor .co.in etc.,)", priority = 5)  public void Invalid4() throws Exception {  beforeTest();  driver.get("https://demowebshop.tricentis.com/register");  driver.manage().window().maximize();  Thread.sleep(5000);  driver.findElement(By.id("gender-male")).click(); driver.findElement(By.name("FirstName")).sendKeys("ffhasdbv@#$%\_-hu");  driver.findElement(By.name("LastName")).sendKeys("fsyaabfj ^&%$!@ sai \_-");  driver.findElement(By.name("Email")).sendKeys("$^$%$$^$&\*#\_-+=");  driver.findElement(By.name("Password")).sendKeys("ertyui");  driver.findElement(By.name("ConfirmPassword")).sendKeys("ertyui");  System.out.print("Test Case - 5");  driver.findElement(By.name("register-button")).click();  driver.findElement(By.xpath("(//input)[5]")).click();  String actual\_message= driver.findElement(By.className("topic-html-content-header")).getText();  String expected\_message="Welcome to our store";  System.out.println(actual\_message);  Assert.assertEquals(actual\_message, expected\_message);  System.out.print("Test Case 5 passed"); }  @AfterTest()  public void afterTest() throws Exception {  Thread.sleep(2000);  driver.quit(); } } |

**RESULT ANALYSIS**

