# ASSIGNMENT

Submitted By,
Romeo Roshan
Roll No. 50

# Selenium, Cucumber, Steps to create automated testing, code(Feature file, Java File)

## Selenium

Selenium is a popular open-source software testing framework that has revolutionized web application testing for developers and quality assurance (QA) professionals. With its suite of powerful tools, Selenium enables you to create and execute automated tests for web applications, allowing you to simulate user interactions with a web application, such as clicking links, filling out forms, and submitting data, while capturing and analysing test results. The framework offers a WebDriver API that automates browser interactions with popular web browsers such as Google Chrome, Firefox, Safari, and Microsoft Edge, making it an indispensable tool for testing web applications.

Selenium offers numerous benefits to software developers and QA professionals, including improved testing efficiency, reduced time-to-market, and higher software quality. With support for multiple programming languages, it provides maximum flexibility and is widely used across the software development industry. Moreover, Selenium is highly customizable and extensible, enabling you to tailor it to your specific testing needs.

Whether you are a seasoned developer or a QA professional, Selenium provides the tools you need to optimize your web application testing process. By automating repetitive tasks and streamlining testing workflows, it helps you achieve better software development outcomes, ensuring that your web applications are robust, reliable, and meet the highest standards of quality.

## Cucumber

Cucumber is an essential tool for teams practicing behavior-driven development (BDD), enabling developers, testers, and business stakeholders to collaborate and define the behavior of a software application in plain English. With its framework for writing and executing automated tests in a natural language format, Cucumber makes it easy for stakeholders with different technical backgrounds to understand and contribute to the testing process.

At the core of Cucumber is the Gherkin language, a domain-specific language (DSL) designed to describe the behaviour of a software application in a structured and easy-to-understand way. Using Gherkin, stakeholders can describe the steps of a scenario using keywords such as Given, When, and Then, ensuring the expected behaviour of an application is defined in a structured and consistent manner.

In combination with Selenium, Cucumber is an incredibly powerful tool for automating web application testing. With Cucumber, you can write tests that simulate user interactions with a web application, including clicking links, filling out forms, and submitting data. Additionally, Cucumber generates reports that offer valuable insights into the testing process, such as the number of passing and failing tests, execution time, and overall test coverage.

By facilitating collaboration and structuring the definition of software application behaviour, Cucumber provides teams with a powerful means of improving software quality and accelerating time-to-market. Its ability to bridge the communication gap between stakeholders with different technical backgrounds makes it a must-have tool for software development teams of all sizes.

## Steps for setting up a Selenium-Cucumber testing framework using

## Eclipse with Firefox driver:

1. Install Eclipse IDE: Download and install the latest version of Eclipse IDE for Java Developers from the Eclipse website.
2. Install Cucumber Eclipse Plugin: Open Eclipse and navigate to Help > Eclipse Marketplace. Search for "Cucumber Eclipse Plugin" and install the plugin.
3. Create a new Maven project: Navigate to File > New > Other > Maven > Maven Project. Select "Create a simple project" and click Next. Enter a Group Id and Artifact Id for your project, and click Finish.
4. Add dependencies to your POM file: Open your project's POM file and add the following dependencies:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">

 <modelVersion>4.0.0</modelVersion>

 <groupId>selenium</groupId>

 <artifactId>selenium</artifactId>

 <version>0.0.1-SNAPSHOT</version>

 <name>testing</name>

 <dependencies>

        <!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-java -->

<dependency>
```

```
    <groupId>io.cucumber</groupId>
    <artifactId>cucumber-java</artifactId>
    <version>7.11.1</version>
</dependency>
<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-junit -->
<dependency>
    <groupId>io.cucumber</groupId>
    <artifactId>cucumber-junit</artifactId>
    <version>7.11.1</version>
    <scope>test</scope>
</dependency>
<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
<dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>4.8.1</version>
</dependency>
<!-- https://mvnrepository.com/artifact/junit/junit -->
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.13.2</version>
    <scope>test</scope>
</dependency>
```

</dependencies>

</project>

5. Create a feature file: Right-click on your project, select New > File. Enter a file name with ".feature" extension and select "Cucumber Feature File" from the dropdown. Write your test scenarios in the feature file.
6. Create step definitions: Right-click on your project, select New > Package. Enter a package name and click Finish. Right-click on the package, select New > Class. Enter a class name and select "JUnit Tes Case" from the dropdown. Write your step definitions in the class.
7. Create a test projectlogin class: Right-click on your project, select New > File. Enter a file name with "projectlogin" suffix and select "JUnit Test Suite" from the dropdown. Write the following code in the test projectlogin class
8. Download and add geckodriver to your project: Download the latest version of geckodriver from the Mozilla website and extract it to a location on your computer. Add the location of the geckodriver executable to your system's PATH variable. In Eclipse, right-click on your project, select New > Folder. Enter "drivers" as the folder name and click Finish. Copy the geckodriver executable to the driver's folder.
9. Run the tests: Right-click on the projectlogin class, select Run As > JUnit Test. The Firefox browser should open and run your tests.

## Creating feature file

To create a feature file in Selenium Cucumber, follow these steps:

1. Open your preferred text editor or Integrated Development Environment (IDE) and create a new file.
2. Save the file with a ".feature" extension, as this is the convention for Cucumber feature files.
3. Write the feature description at the top of the file. This should describe the high-level functionality of the feature you are testing.
4. Define the scenarios using the Gherkin syntax. The scenarios should describe the steps required to test the feature, using the Given, When, and Then keywords.

Feature: Test login functionality

Scenario: check login is successful with valid credential

Given browser is open

And user is on login page

When user enter username and password

And user clicks on login

Then user is navigated to the home page

5. Save the feature file in the appropriate location within your project directory.

Once you have created the feature file, you can create the corresponding step definitions in your preferred programming language (e.g., Java, Python) to automate the scenarios using Selenium WebDriver.

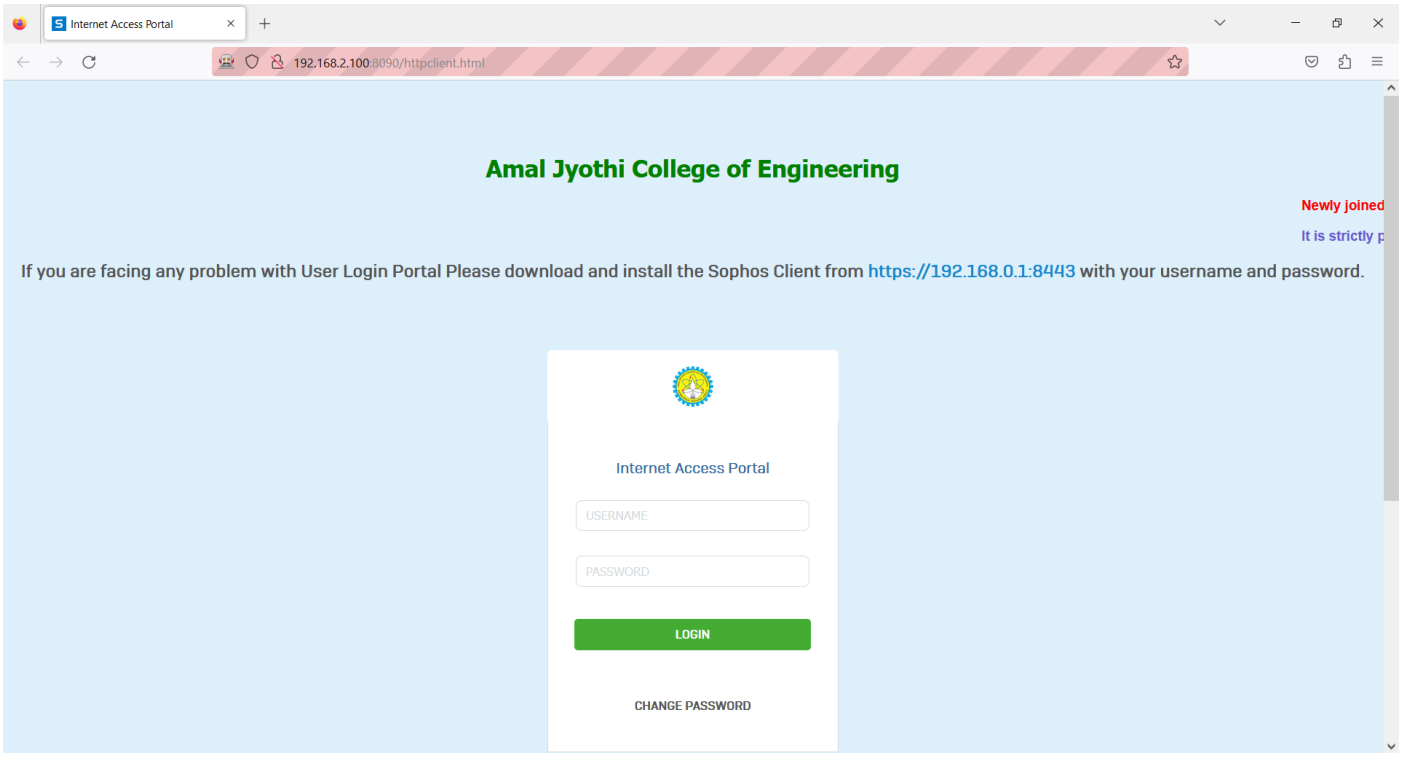
## Creating java file

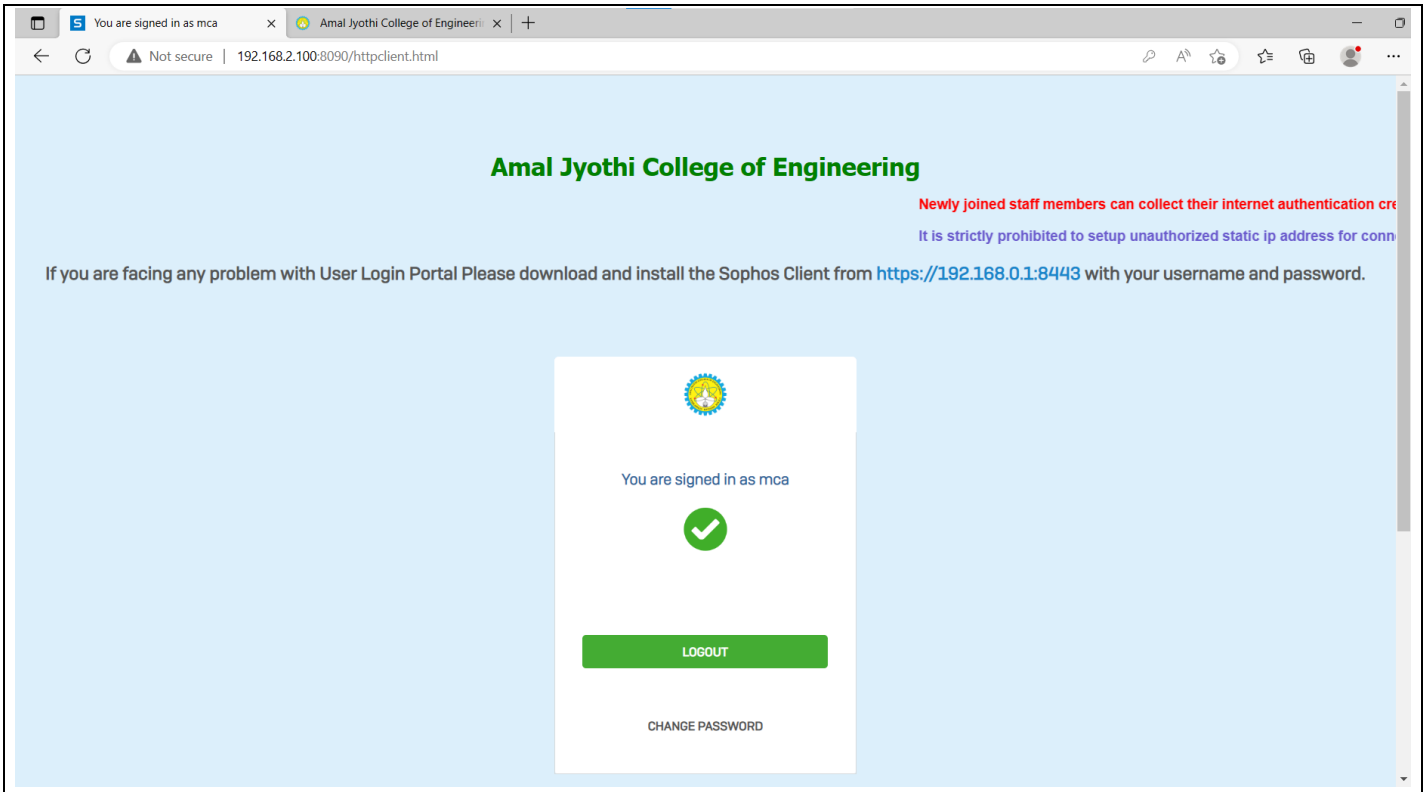
```
package stepdefinition;


import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;


import io.cucumber.java.en.And;

import io.cucumber.java.en.Given;

import io.cucumber.java.en.Then;

import io.cucumber.java.en.When;


public class loginsteps {

	WebDriver driver=null;

	@Given("browser is open")

	public void browser_is_open() {


	 System.setProperty("webdriver.geko.marionette","C:\\Users\\ROG STRIX\\eclipse-workspace\\selenium\\src\\test\\resources\\files\\geckodriver.exe");

	 driver=new FirefoxDriver();
```

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

@And("user is on login page")
public void user_is_on_login_page()  throws Exception{
  driver.navigate().to("http://192.168.2.100:8090/httpclient.html");
  Thread.sleep(2000);

}

@When("user enters username and password")
public void user_enters_username_and_password()  throws Throwable{
  driver.findElement(By.id("username")).sendKeys("mca");
  driver.findElement(By.id("password")).sendKeys("mca");

}

@And("User clicks on login")
public void user_clicks_on_login() {
  driver.findElement(By.id("loginbutton")).click();

}

@Then("user is navigated to the home page")
public void user_is_navigated_to_the_home_page() throws Exception{
```

```
        driver.findElement(By.id("signin-caption")).isDisplayed();
        Thread.sleep(2000);
        driver.close();
        driver.quit();

    }

}
```

# Final Output




You are signed in as mca
Amal Jyothi College of Engineeri
Not secure | 192.168.2.100:8090/httpclient.html
Amal Jyothi College of Engineering
Newly joined staff members can collect their internet authentication cre
It is strictly prohibited to setup unauthorized static ip address for conn
If you are facing any problem with User Login Portal Please download and install the Sophos Client from https://192.168.0.1:8443 with your username and password.
You are signed in as mca
LOGOUT
CHANGE PASSWORD