srirampura, bengaluru

marathahalli, bangalore

bangalore - marathahalli

plot no 38, bangalore - marathahalli

Pizzahut:

👉Chrome Driver:

[Chrome for Testing availability](https://googlechromelabs.github.io/chrome-for-testing/)

👉Use Install new software in Eclipse to install the plugins

**👉 Cucumber eclipse update site:**

<https://cucumber.github.io/cucumber-eclipse/update-site/>

<https://github.com/cucumber/cucumber-eclipse-update-site>

**👉 TestNG Eclipse update site:**

<https://testng.org/testng-eclipse-update-site/>

**👉 websocket issue: FIX= for older version of selenium dependencies**

ChromeOptions options = new ChromeOptions();

options.addArguments("--remote-allow-origins=\*");

**OR,**

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-http-jdk-client</artifactId>

<version>4.5.0</version>

</dependency>

and added this code line at the first line into @BeforeTest method:

System.setProperty("webdriver.http.factory", "jdk-http-client");

👉Extent Report - Grasshopper tech

<https://mvnrepository.com/artifact/tech.grasshopper/extentreports-cucumber7-adapter>

<!-- https://mvnrepository.com/artifact/tech.grasshopper/extentreports-cucumber7-adapter -->

<dependency>

<groupId>tech.grasshopper</groupId>

<artifactId>extentreports-cucumber7-adapter</artifactId>

<version>1.14.0</version>

</dependency>

👉Selenium Java

<https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java>

<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>4.4.0</version>

</dependency>

👉TestNG

<https://mvnrepository.com/artifact/org.testng/testng>

<!-- https://mvnrepository.com/artifact/org.testng/testng -->

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>6.14.3</version>

<scope>test</scope>

</dependency>

👉Cucumber JUnit

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

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

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-junit</artifactId>

<version>6.8.0</version>

<scope>test</scope>

</dependency>

👉Cucumber - java

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

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

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-java</artifactId>

<version>6.8.0</version>

</dependency>

👉WebDriver Manager - bony garcia

<https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager>

<!-- https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager -->

<dependency>

<groupId>io.github.bonigarcia</groupId>

<artifactId>webdrivermanager</artifactId>

<version>6.1.0</version>

</dependency>

👉 To pause dynamic collapsing of dropdown or any other element

* use a script in the console pause the javascript debugger in a delayed time
* **Method**:
  + setTimeout(() => { debugger; }, 5000)
  + **Without arrow/lambda function**  
    setTimeout(function() {

debugger;

}, 2000);

* Or use the command pallet using CTRL SHIFT P > disable javascript
  + Enable it using enable javascript (when done)

👉Checklists:

1. Build the code all at once and individually.
   1. write all the test cases
2. put them into cucumber test case
3. build their page objects
4. build reportings
   1. use listeners
   2. use extent reports
5. Run with junit test runner

👉 check

1. How to load the chromeDriver with selectorshub capability?

👉Approach:

1. **User launch Pizzahut application with "<URL>"**
   1. Launch using the URL with the webdriver
2. **User wait for auto location black pop up screen**
   1. It pops up when test is going, normal opening of webpage will not show that very often
   2. Location pop up will show the nearest location
   3. it will have a close button
   4. wait for it to load
3. **User close the pop up screen**
   1. wait for 1 -2 sec and then use close element to close
4. **User see pop up for delivery asking for enter location**
   1. It’s the pop up on the main page
   2. it has some tabs
   3. it has a delivery tab
   4. print the delivery tab (in general to validate that the user is on the delivery tab that asks for location to enter)
   5. end case: apply assertion to validate it
5. **User type address as "<Location>"**
   1. Get the location element
   2. send the delivery location to enter the location in the box
   3. If any other cases
6. **User select first auto populate drop down option**
   1. Delivery location box > list down the locations
   2. Select the first location in the list
      1. use loop to iterate and conditions
      2. Select the first one
7. **User navigate to deails page**
   1. The previous action will automatically navigate the user to the Deals tab
   2. Print the deals tab element text to validate
   3. End case: apply assertion to validate that the user is on the deals tab.
8. **User validate vegetarian radio button flag is off**
   1. find the radio button element selector
   2. Valiidate if it’s in the off state
   3. print it
9. **User clicks on Pizzas menu bar option**
   1. Find the element selector of the Pizzas tab
   2. click onto it to navigate to this tab
   3. print the tab text to validate
   4. End case: apply assertion to validate that the user is on the deals tab.
10. **User select add button of any pizza from Recommended**
    1. all the pizzas are in container
    2. Add button is common to all of the pizza containers
    3. use loop to iterate through the container list
    4. use conditional loop to find out the add button for a specific pizza (container)
    5. Click that add button
11. **User see that the pizza is getting added under Your Basket**
    1. After clicking the add button > that specific pizza will get added to the form summary section under the Your basket section
    2. use conditional loop (s) with contains() method to print the text of the pizza under the your basket section and validate/match the text
    3. end case: apply assertion
12. **User validate pizza price plus Tax is checkout price**
    1. Subtotal+ restaurant handling charges + Total tax = checkout price
       1. find subtotal element selector
       2. find handling charges selector
       3. Total tax element selector
       4. calculate the total price
    2. Find the element selector of checkout total price
    3. use conditions to match them both
    4. print if true
    5. End case: apply assertion
13. **User validate checkout button contains Item count**
    1. Find the item counter element selector
    2. Use conditions to find out with contains() method
    3. print it to validate
    4. End case: apply assertion
14. **User validate checkout button contains total price count**
    1. Find the checkout total price element selector (get from previous case)
    2. use conditions with contains() method to validate if the btn contains the total price count
    3. print it to validate > totalPriceOfPizza (variable)
    4. end Case: apply assertion
15. **User clicks on Drinks option**
    1. find Drinks tab element selector
    2. Click it
    3. print the tab text to validate
    4. End case: Apply assertion
16. **User select Pepsi option to add into the Basket**
    1. all the Drinks are in container
    2. Add button is common to all of the Drinks containers
    3. use loop to iterate through the container list
    4. use conditional loop to find out the add button for a specific drink (pepsi)
    5. Click that add button
17. **User see 2 items are showing under checkout button**
    1. Find the chckt btn element selector (get from previous step)
    2. print out the item count = 2 to validate
    3. end case: Apply assertion
18. **User see total price is now more than before**
    1. find the total price element selector from check out button
    2. print it > totalPriceAfterAddingPepsi
    3. use conditions to compare them
    4. Print the result if true or false
19. **User remove the Pizza item from Basket**
    1. Find the element selector for pizza item close button/remove button
    2. Click it
    3. print the container list text to validate that that the pizza has been removed
20. **see Price tag got removed from the checkout button**
    1. find the checkout button element selector
    2. use conditions with contains() method to find out if the price tag has been removed from the button
    3. end case: apply assertion
21. **User see 1 item showing in checkout button**
    1. find the checkout button element selector
    2. Use conditions with contains() to check out if the item counter is showing 1 item
    3. End case: apply assertion
22. **User Clicks on Checkout button**
    1. find the checkout button element selector
    2. click it
23. **User see minimum order required pop up is getting displayed**
    1. A pop up will appear which contains “Minimum Delivery” text
    2. use conditions with contains to compare the text with the given text
    3. print the whole text
    4. End case: Apply assertion

👉 **Test Scenario 2:[TestNG+Selenium]**

1.**Create a testNG.xml file in the same Maven project**

1. use new file to create it

2.**Configure a test pizzahut001 inside testNG.xml file**

1. configure the test suite

3.**Create a new java file for TestNG test implementation**

1. test case file
2. define all the test cases using @Test
3. A completely different test case file to implement using testNG
4. use priority to the test case methods to prioritize the steps serially

4.**Use Extend Reporting while each step is PASS/FAIL**

5.**Use an Excel Sheet to read application URL [**[**https://www.pizzahut.co.in/**](https://www.pizzahut.co.in/)**]**

6.**TestNG test steps are given below:**

✅ Use @BeforeClass/@BeforeSuite to launch https://www.pizzahut.co.in/

* A black color auto pop up screen will be displayed. Close it
* Then, set the user delivery location as Lulu Mall, Bangalore
* The user is now on the Deals page. Validate that the URL has text as ‘deals’
* Go to sides and add any item that is below 200
* Validate that the product is added under Basket but checkout button price item is still now showing
* Navigate to the Drinks page
* Add any two drinks so that total cart pricing is more than 200
* Click on the Checkout button. The user will be navigated to the checkout page
* Validate that the Online Payment radio button is selected by default
* Change the Payment option to Cash
* Validate that the I agree checkbox is checked by default
* Enter name, mobile, and email address
* Click on the Apply Gift Card link
* A pop up should appear. Click on the Voucher
* Give the Voucher code as 12345 and submit
* Validate if an error is coming that the number is incorrect
* Close the voucher pop up
* The user should again navigate to your Basket page.

✅ Use @AfterClass/@AfterSuite to quit the driver session

👉Configure testNG.xml to run pom.xml

👉All test data should be parameterized from testNG.xml

👉Use page object model to capture page object

👉Upload the project into Git

👉Make sure the project is running using the Maven test command