**Buy\_food\_item.feature (source code)**

Feature: a user should able to buy food item on Swiggy

@TC101 @regression @pizza

Scenario: Buy food item

Given a user is on the landing page of Swiggy

When he enters location as chennai in the search bar

And he presses the down-arrow key then hit the enter button

And he clicks on the search icon

And he enters food item name in the searchbox as "<fooditem>"

And he selects the first item in the auto complete box

And he clicks on the add button to add the first item

And he clicks on the add item button from the popup

And he hovers on the cart icon

And he clicks on the checkout button

Then he able to buy the food item successfully

Examples:

|fooditem|

|pizza|

**BuyFoodItemSteps.java(source code)**

**package** com.swiggy.testscripts;

**import** org.testng.Assert;

**import** io.cucumber.java.en.Given;

**import** io.cucumber.java.en.Then;

**import** io.cucumber.java.en.When;

**public** **class** BuyFoodItemSteps **extends** Driver{

@Given("a user is on the landing page of Swiggy")

**public** **void** a\_user\_is\_on\_the\_landing\_page\_of\_swiggy() {

//Verifying the user is at landing page

Assert.*assertTrue*(*driver*.getTitle().equals("Order food online from India's best food delivery service. Order from restaurants near you"));

}

@When("he enters location as chennai in the search bar")

**public** **void** he\_enters\_location\_as\_chennai\_in\_the\_search\_bar() **throws** InterruptedException {

//he enters location as chennai in the location searchbar

*landingPage*.enterLocation();

Thread.*sleep*(4000);

}

@When("he press the down-arrow key then hit the enter button")

**public** **void** he\_press\_the\_down\_arrow\_key\_then\_hit\_the\_enter\_button() {

//he select the dirst option from the auto-complete box

*landingPage*.keyBoardOperation();

}

@When("he clicks on the search icon")

**public** **void** he\_clicks\_on\_the\_search\_icon() {

//he clicks on the search icon to enter food name

*searchFood*.clickOnSearchIcon();

}

@When("he enter food item name in the searchbox as {string}")

**public** **void** he\_enter\_food\_item\_name\_in\_the\_searchbox\_as(String string) {

//he enters the food name in the search-bar

*enterFood*.enterFoodName(string);

}

@When("he select the first item in the auto complete box")

**public** **void** he\_select\_the\_first\_item\_in\_the\_auto\_complete\_box() **throws** InterruptedException {

//he select the first food item from the auto complete box

Thread.*sleep*(4000);

*enterFood*.keyBoardOperation1();

}

@When("he click on the add button to add the first item")

**public** **void** he\_click\_on\_the\_add\_button\_to\_add\_the\_first\_item() {

//he clicks on the add button to add the item into the cart

*foodList*.clickOnAddBtn();

}

@When("he click on the add item button from the popup")

**public** **void** he\_click\_on\_the\_add\_item\_button\_from\_the\_popup() {

//click on add Item on the popup

*foodList*.clickOnAddItem();

}

@When("he hovers on the cart icon")

**public** **void** he\_hovers\_on\_the\_cart\_icon() **throws** InterruptedException {

//he hovers over the cart icon

Thread.*sleep*(4000);

*foodList*.hoverOverCartIcon();

}

@When("he click on the checkout button")

**public** **void** he\_click\_on\_the\_checkout\_button() {

//he clicks on the checkout button

*foodList*.clickOnCheckOutBtn();

}

@Then("he able to buy the food item successfully")

**public** **void** he\_able\_to\_buy\_the\_food\_item\_successfully() {

String expText="SECURE CHECKOUT";

String actText= *buyFood*.getTheText();

Assert.*assertEquals*(actText, expText);

}

**Tools.java(source code)**

**package** com.swiggy.testscripts;

**import** org.openqa.selenium.WebDriver;

**public** **class** Tools {

**protected** **static** WebDriver *driver*;

}

**Driver.java(source code)**

package com.swiggy.testscripts;

import org.openqa.selenium.chrome.ChromeDriver;

import com.swiggy.pages.BuyFood;

import com.swiggy.pages.EnterFood;

import com.swiggy.pages.FoodList;

import com.swiggy.pages.LandingPage;

import com.swiggy.pages.SearchFood;

public class Driver extends Tools{

protected static LandingPage landingPage;

protected static SearchFood searchFood;

protected static EnterFood enterFood;

protected static FoodList foodList;

protected static BuyFood buyFood;

public static void init()

{

System.setProperty("webdriver.chrome.driver", "chromedriver.exe");

driver= new ChromeDriver();

landingPage=new LandingPage(driver);

searchFood=new SearchFood(driver);

enterFood=new EnterFood(driver);

foodList=new FoodList(driver);

buyFood=new BuyFood(driver);

}

}

**BeforeAfter.java(source code)**

**package** com.swiggy.testscripts;

**import** io.cucumber.java.After;

**import** io.cucumber.java.Before;

**import** io.cucumber.java.Scenario;

**public** **class** BeforeAfter **extends** Tools{

@Before

**public** **void** setUp(Scenario scenario)

{

Driver.*init*();

*driver*.manage().window().maximize();

*driver*.get("https://www.swiggy.com");

}

@After

**public** **void** tearDown()

{

*driver*.quit();

}

}

**TestRunner.java(source code)**

**package** com.swiggy.testscripts;

**import** io.cucumber.testng.AbstractTestNGCucumberTests;

**import** io.cucumber.testng.CucumberOptions;

@CucumberOptions(features="Feature",

glue="com.swiggy.testscripts",

tags="@regression and @pizza"

)

**public** **class** TestRunner **extends** AbstractTestNGCucumberTests{

}

**LandingPage.java(source code)**

**package** com.swiggy.pages;

**import** org.openqa.selenium.Keys;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

**import** org.openqa.selenium.interactions.Actions;

**import** org.openqa.selenium.support.FindBy;

**import** org.openqa.selenium.support.PageFactory;

**public** **class** LandingPage {

**private** Actions actions;

**public** LandingPage(WebDriver driver)

{

PageFactory.*initElements*(driver, **this**);

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

}

@FindBy(id="location")

**private** WebElement locationSearchBar;

**public** **void** enterLocation()

{

locationSearchBar.sendKeys("Chennai");

}

**public** **void** keyBoardOperation()

{

actions.sendKeys(Keys.***ARROW\_DOWN***).sendKeys(Keys.***ENTER***).build().perform();

}

}

**SearchFood.java(source code)**

**package** com.swiggy.pages;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

**import** org.openqa.selenium.support.FindBy;

**import** org.openqa.selenium.support.PageFactory;

**import** org.openqa.selenium.support.ui.ExpectedConditions;

**import** org.openqa.selenium.support.ui.WebDriverWait;

**public** **class** SearchFood {

**private** WebDriverWait wait;

**public** SearchFood(WebDriver driver)

{

PageFactory.*initElements*(driver, **this**);

wait = **new** WebDriverWait(driver, 10);

}

@FindBy(xpath="(//div[contains(@class,'\_2CgXb')])[5]")

**private** WebElement searchicon;

**public** **void** clickOnSearchIcon()

{

wait.until(ExpectedConditions.*visibilityOf*(searchicon));

searchicon.click();

}

}

**EnterFood.java(source code)**

**package** com.swiggy.pages;

**import** org.openqa.selenium.Keys;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

**import** org.openqa.selenium.interactions.Actions;

**import** org.openqa.selenium.support.FindBy;

**import** org.openqa.selenium.support.PageFactory;

**import** org.openqa.selenium.support.ui.ExpectedConditions;

**import** org.openqa.selenium.support.ui.WebDriverWait;

**public** **class** EnterFood {

**private** Actions actions;

**private** WebDriverWait wait;

**public** EnterFood(WebDriver driver)

{

PageFactory.*initElements*(driver, **this**);

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

wait = **new** WebDriverWait(driver, 10);

}

@FindBy(xpath="//input[contains(@class,'\_2BJMh')]")

**private** WebElement foodSearchBar;

**public** **void** enterFoodName(String food)

{

wait.until(ExpectedConditions.*visibilityOf*(foodSearchBar));

foodSearchBar.clear();

foodSearchBar.sendKeys(food);

}

**public** **void** keyBoardOperation1()

{

//wait.until(ExpectedConditions.visibilityOf(foodSearchBar));

actions.sendKeys(Keys.***ARROW\_DOWN***).sendKeys(Keys.***ENTER***).build().perform();

}

}

**FoodList.java(source code)**

**package** com.swiggy.pages;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

**import** org.openqa.selenium.interactions.Actions;

**import** org.openqa.selenium.support.FindBy;

**import** org.openqa.selenium.support.PageFactory;

**import** org.openqa.selenium.support.ui.ExpectedConditions;

**import** org.openqa.selenium.support.ui.WebDriverWait;

**public** **class** FoodList {

**private** Actions actions;

**private** WebDriverWait wait;

**public** FoodList(WebDriver driver)

{

PageFactory.*initElements*(driver, **this**);

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

wait = **new** WebDriverWait(driver, 10);

}

@FindBy(xpath="(//div[contains(@class,'\_1RPOp')])[1]")

**private** WebElement addButton;

**public** **void** clickOnAddBtn()

{

wait.until(ExpectedConditions.*visibilityOf*(addButton));

addButton.click();

}

@FindBy(xpath="//div[contains(@class,'\_1fmVk \_30y3a')]")

**private** WebElement cartIcon;

**public** **void** hoverOverCartIcon()

{

actions.moveToElement(cartIcon).build().perform();

}

@FindBy(xpath="//div[contains(@class,'\_55uP6')]")

**private** WebElement checkOutBtn;

**public** **void** clickOnCheckOutBtn()

{

wait.until(ExpectedConditions.*visibilityOf*(checkOutBtn));

checkOutBtn.click();

}

@FindBy(xpath="//div[contains(@class,'\_3coNr')]")

**private** WebElement addItemBtn;

**public** **void** clickOnAddItem()

{

wait.until(ExpectedConditions.*visibilityOf*(addItemBtn));

addItemBtn.click();

}

}

**BuyFood.java(source code)**

**package** com.swiggy.pages;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

**import** org.openqa.selenium.support.FindBy;

**import** org.openqa.selenium.support.PageFactory;

**import** org.openqa.selenium.support.ui.ExpectedConditions;

**import** org.openqa.selenium.support.ui.WebDriverWait;

**public** **class** BuyFood {

**private** WebDriverWait wait;

**public** BuyFood(WebDriver driver)

{

PageFactory.*initElements*(driver, **this**);

wait = **new** WebDriverWait(driver, 10);

}

@FindBy(xpath="//div[contains(@class,'B5DgV')]")

**private** WebElement text;

**public** String getTheText()

{

wait.until(ExpectedConditions.*visibilityOf*(text));

String Lasttext= text.getText();

**return** Lasttext;

}

}

**ExtentManager.java(source code)**

package com.listeners;

import java.io.File;

import com.aventstack.extentreports.ExtentReports;

import com.aventstack.extentreports.reporter.ExtentHtmlReporter;

import com.aventstack.extentreports.reporter.configuration.ChartLocation;

import com.aventstack.extentreports.reporter.configuration.Theme;

public class ExtentManager {

private static ExtentReports extent;

private static String reportFileName = "Test-Automation-Report"+".html";

private static String fileSeperator = System.getProperty("file.separator");

private static String reportFilepath = System.getProperty("user.dir") +fileSeperator+ "TestReport";

private static String reportFileLocation = reportFilepath +fileSeperator+ reportFileName;

public static ExtentReports getInstance() {

if (extent == null)

createInstance();

return extent;

}

//Create an extent report instance

public static ExtentReports createInstance() {

String fileName = getReportPath(reportFilepath);

ExtentHtmlReporter htmlReporter = new ExtentHtmlReporter(fileName);

htmlReporter.config().setTestViewChartLocation(ChartLocation.TOP);

htmlReporter.config().setChartVisibilityOnOpen(true);

htmlReporter.config().setTheme(Theme.STANDARD);

htmlReporter.config().setDocumentTitle(reportFileName);

htmlReporter.config().setEncoding("utf-8");

htmlReporter.config().setReportName(reportFileName);

htmlReporter.config().setTimeStampFormat("EEEE, MMMM dd, yyyy, hh:mm a '('zzz')'");

extent = new ExtentReports();

extent.attachReporter(htmlReporter);

//Set environment details

extent.setSystemInfo("OS", "Windows");

extent.setSystemInfo("AUT", "QA");

return extent;

}

//Create the report path

private static String getReportPath (String path) {

File testDirectory = new File(path);

if (!testDirectory.exists()) {

if (testDirectory.mkdir()) {

System.out.println("Directory: " + path + " is created!" );

return reportFileLocation;

} else {

System.out.println("Failed to create directory: " + path);

return System.getProperty("user.dir");

}

} else {

System.out.println("Directory already exists: " + path);

}

return reportFileLocation;

}

}

**ExtentTestManager.java(source code)**

package com.listeners;

import java.util.HashMap;

import java.util.Map;

import com.aventstack.extentreports.ExtentReports;

import com.aventstack.extentreports.ExtentTest;

public class ExtentTestManager {

static Map<Integer, ExtentTest> extentTestMap = new HashMap<Integer, ExtentTest>();

static ExtentReports extent = ExtentManager.getInstance();

public static synchronized ExtentTest getTest() {

return (ExtentTest) extentTestMap.get((int) (long) (Thread.currentThread().getId()));

}

public static synchronized void endTest() {

extent.flush();

}

public static synchronized ExtentTest startTest(String testName) {

ExtentTest test = extent.createTest(testName);

extentTestMap.put((int) (long) (Thread.currentThread().getId()), test);

return test;

}

}

**TestListener.java(source code)**

package com.listeners;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.text.SimpleDateFormat;

import org.openqa.selenium.OutputType;

import org.openqa.selenium.TakesScreenshot;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.io.FileHandler;

import org.testng.ITestContext;

import org.testng.ITestListener;

import org.testng.ITestResult;

import com.aventstack.extentreports.MediaEntityBuilder;

import com.aventstack.extentreports.Status;

import com.aventstack.extentreports.model.Log;

import org.apache.log4j.Logger;

public class TestListener implements ITestListener {

private static Logger log = Logger.getLogger(Log.class.getName());

public void onStart(ITestContext context) {

System.out.println("\*\*\* Test Suite " + context.getName() + " started \*\*\*");

}

public void onFinish(ITestContext context) {

System.out.println(("\*\*\* Test Suite " + context.getName() + " ending \*\*\*"));

ExtentTestManager.endTest();

ExtentManager.getInstance().flush();

}

public void onTestStart(ITestResult result) {

System.out.println(("\*\*\* Running test method " + result.getMethod().getMethodName() + "..."));

ExtentTestManager.startTest(result.getMethod().getMethodName());

}

public void onTestSuccess(ITestResult result) {

System.out.println("\*\*\* Executed " + result.getMethod().getMethodName() + " test successfully...");

ExtentTestManager.getTest().log(Status.PASS, "Test passed");

}

public void onTestFailure(ITestResult result) {

log.info("\*\*\* Test execution " + result.getMethod().getMethodName() + " failed...");

log.info((result.getMethod().getMethodName() + " failed!"));

//ITestContext context = result.getTestContext();

WebDriver driver = (WebDriver) result.getTestContext().getAttribute("driver");

String targetLocation = null;

String testClassName = result.getInstanceName().trim();

String timeStamp = new SimpleDateFormat("yyyy.MM.dd.HH.mm.ss").format(new java.util.Date()); // get timestamp

String testMethodName = result.getName().toString().trim();

String screenShotName = testMethodName + timeStamp + ".png";

String fileSeperator = System.getProperty("file.separator");

String reportsPath = System.getProperty("user.dir") + fileSeperator + "TestReport" + fileSeperator

+ "screenshots";

log.info("Screen shots reports path - " + reportsPath);

try {

File file = new File(reportsPath + fileSeperator + testClassName); // Set

// screenshots

// folder

if (!file.exists()) {

if (file.mkdirs()) {

log.info("Directory: " + file.getAbsolutePath() + " is created!");

} else {

log.info("Failed to create directory: " + file.getAbsolutePath());

}

}

File screenshotFile = ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);

targetLocation = reportsPath + fileSeperator + testClassName + fileSeperator + screenShotName;// define

// location

File targetFile = new File(targetLocation);

log.info("Screen shot file location - " + screenshotFile.getAbsolutePath());

log.info("Target File location - " + targetFile.getAbsolutePath());

FileHandler.copy(screenshotFile, targetFile);

} catch (FileNotFoundException e) {

log.info("File not found exception occurred while taking screenshot " + e.getMessage());

} catch (Exception e) {

log.info("An exception occurred while taking screenshot " + e.getCause());

}

// attach screenshots to report

try {

ExtentTestManager.getTest().fail("Screenshot",

MediaEntityBuilder.createScreenCaptureFromPath(targetLocation).build());

} catch (IOException e) {

log.info("An exception occured while taking screenshot " + e.getCause());

}

ExtentTestManager.getTest().log(Status.FAIL, "Test Failed");

}

public void onTestSkipped(ITestResult result) {

System.out.println("\*\*\* Test " + result.getMethod().getMethodName() + " skipped...");

ExtentTestManager.getTest().log(Status.SKIP, "Test Skipped");

}

public void onTestFailedButWithinSuccessPercentage(ITestResult result) {

System.out.println("\*\*\* Test failed but within percentage % " + result.getMethod().getMethodName());

}

}

**pom.xml**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<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 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>com.cisco</groupId>

<artifactId>Swiggy</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>Swiggy</name>

<!-- FIXME change it to the project's website -->

<url>http://www.example.com</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<maven.compiler.source>1.7</maven.compiler.source>

<maven.compiler.target>1.7</maven.compiler.target>

</properties>

<dependencies>

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

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-core</artifactId>

<version>7.0.0</version>

</dependency>

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

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-java</artifactId>

<version>7.0.0</version>

</dependency>

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

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-testng</artifactId>

<version>7.0.0</version>

</dependency>

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

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-jvm-deps</artifactId>

<version>1.0.6</version>

<scope>provided</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/net.masterthought/cucumber-reporting -->

<dependency>

<groupId>net.masterthought</groupId>

<artifactId>cucumber-reporting</artifactId>

<version>5.6.1</version>

</dependency>

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

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>gherkin</artifactId>

<version>22.0.0</version>

</dependency>

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

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>7.4.0</version>

</dependency>

<!-- https://mvnrepository.com/artifact/com.beust/jcommander -->

<dependency>

<groupId>com.beust</groupId>

<artifactId>jcommander</artifactId>

<version>1.81</version>

</dependency>

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

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-html</artifactId>

<version>0.2.7</version>

</dependency>

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

<dependency>

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

<artifactId>selenium-java</artifactId>

<version>3.141.59</version>

</dependency>

<dependency>

<groupId>com.aventstack</groupId>

<artifactId>extentreports</artifactId>

<version>3.1.5</version>

</dependency>

<dependency>

<groupId>org.apache.logging.log4j</groupId>

<artifactId>log4j-api</artifactId>

<version>2.11.1</version>

</dependency>

<dependency>

<groupId>org.apache.logging.log4j</groupId>

<artifactId>log4j-core</artifactId>

<version>2.11.1</version>

</dependency>

<dependency>

<groupId>org.apache.logging.log4j</groupId>

<artifactId>log4j-1.2-api</artifactId>

<version>2.11.1</version>

</dependency>

</dependencies>

<build>

<pluginManagement><!-- lock down plugins versions to avoid using Maven defaults (may be moved to parent pom) -->

<plugins>

<!-- clean lifecycle, see https://maven.apache.org/ref/current/maven-core/lifecycles.html#clean\_Lifecycle -->

<plugin>

<artifactId>maven-clean-plugin</artifactId>

<version>3.1.0</version>

</plugin>

<!-- default lifecycle, jar packaging: see https://maven.apache.org/ref/current/maven-core/default-bindings.html#Plugin\_bindings\_for\_jar\_packaging -->

<plugin>

<artifactId>maven-resources-plugin</artifactId>

<version>3.0.2</version>

</plugin>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.0</version>

</plugin>

<plugin>

<artifactId>maven-surefire-plugin</artifactId>

<version>2.22.1</version>

</plugin>

<plugin>

<artifactId>maven-jar-plugin</artifactId>

<version>3.0.2</version>

</plugin>

<plugin>

<artifactId>maven-install-plugin</artifactId>

<version>2.5.2</version>

</plugin>

<plugin>

<artifactId>maven-deploy-plugin</artifactId>

<version>2.8.2</version>

</plugin>

<!-- site lifecycle, see https://maven.apache.org/ref/current/maven-core/lifecycles.html#site\_Lifecycle -->

<plugin>

<artifactId>maven-site-plugin</artifactId>

<version>3.7.1</version>

</plugin>

<plugin>

<artifactId>maven-project-info-reports-plugin</artifactId>

<version>3.0.0</version>

</plugin>

</plugins>

</pluginManagement>

</build>

</project>

**Testing.xml**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name=*"Suite"*>

<listeners>

<listener class-name=*"com.listeners.TestListener"*></listener>

</listeners>

<test thread-count=*"5"* name=*"Test"*>

<classes>

<class name=*"com.swiggy.testscripts.TestRunner"*></class>

</classes>

</test> <!-- Test -->

</suite> <!-- Suite -->