Create a Cucumber Framework for Swiggy App – Source code

Verify_text.feature

```
Feature: Checkout page Text

  Scenario: A user can see checkout page text
    Given a user is on the landing page
    When he type the location and clicks 1st location
    And he click 1st restaurant under Top restaurant chains in Hyderabad
    And he clicks add button on first listed dish
    And hover over the cart icon in right top corner
    And he clicks Checkout button
    Then verify the text "To place your order now, log in to your existing
account or sign up."
```

src/main/java

LandingPage.java

```
package com.Swiggy.Pages;

import java.time.Duration;
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 LandingPage {
	private Actions actions;
	private WebDriverWait wait;

	@FindBy(id = "location")
	private WebElement locationField;

	@FindBy(className = "_1oLDb")
	private WebElement autoCompleteBox;

	public LandingPage(WebDriver driver) {
		PageFactory.initElements(driver, this);
		actions = new Actions(driver);
		wait = new WebDriverWait(driver, Duration.ofSeconds(60));
	}

	public void enterlLocation(String location) {
		locationField.sendKeys(location);
```

```
        wait.until(ExpectedConditions.visibilityOfAllElements(autoCompleteBo
x));

        actions.sendKeys(Keys.ARROW_DOWN).sendKeys(Keys.ENTER).build().perfo
rm();
        }
}
```

SearchResultsRestaurant.java

```
package com.Swiggy.Pages;

import java.time.Duration;
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 SearchResultsRestaurant {

        private WebDriverWait wait;

        @FindBy(xpath = "//div[contains(@class, 'gWiXXg')][1]")
        private WebElement firstRestaurant;

        public SearchResultsRestaurant(WebDriver driver) {
                PageFactory.initElements(driver, this);
                wait = new WebDriverWait(driver, Duration.ofSeconds(60));
        }

        public void clickFirstRestaurant() {

        wait.until(ExpectedConditions.visibilityOfAllElements(firstRestauran
t));
                firstRestaurant.click();
        }
}
```

DishesAndCartPage.java

```
package com.Swiggy.Pages;

import java.time.Duration;
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 DishesAndCartPage {
	private Actions actions;
	private WebDriverWait wait;

	@FindBy(xpath = "//div[contains(@class, '_1RPOp')][1]")
	private WebElement firstDish;

	@FindBy(xpath = "//span[contains(@class, '_2vS77')]")
	private WebElement cartIcon;

	@FindBy(xpath = "//div[contains(@class, '_55uP6')]")
	private WebElement checkoutBtn;

	public DishesAndCartPage(WebDriver driver) {
		PageFactory.initElements(driver, this);
		actions = new Actions(driver);
		wait = new WebDriverWait(driver, Duration.ofSeconds(60));
	}

	public void clickFirstRestaurant() {

	wait.until(ExpectedConditions.visibilityOfAllElements(firstDish));
		firstDish.click();
	}

	public void hoverOverCartIcon() throws InterruptedException {
		Thread.sleep(4000);

	wait.until(ExpectedConditions.visibilityOfAllElements(cartIcon));
		actions.moveToElement(cartIcon).build().perform();
	}

	public void clickCheckout() {
		wait.until(ExpectedConditions.visibilityOf(checkoutBtn));
		checkoutBtn.click();
	}
}
```

CheckoutPage.java

```
package com.Swiggy.Pages;

import java.time.Duration;
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 CheckoutPage {
	private WebDriverWait wait;

	@FindBy(xpath = "//div[contains(@class, '_2axtv')]")
	private WebElement verificationText;
	public CheckoutPage(WebDriver driver) {
		PageFactory.initElements(driver, this);
		wait = new WebDriverWait(driver, Duration.ofSeconds(60));
	}

	public String verifyText() {
		wait.until(ExpectedConditions.visibilityOf(verificationText));
		String message = verificationText.getText();
		return message;
	}

}
```

src/test/java

ExtentManager.java

```
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

```
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();

	@SuppressWarnings("deprecation")
	public static synchronized ExtentTest getTest() {
		return (ExtentTest) extentTestMap.get((int) (long)
(Thread.currentThread().~~getId~~()));
```

```
	}

	public static synchronized void endTest() {
		extent.flush();
	}

	@SuppressWarnings("deprecation")
	public static synchronized ExtentTest startTest(String testName) {
		ExtentTest test = extent.createTest(testName);
		extentTestMap.put((int) (long)
(Thread.currentThread().~~getId~~()), test);
		return test;
	}

}
```

TestListener.java

```
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());
      }
}
```

BeforeAfter.java

```
package com.Swiggy.TestSteps;

import io.cucumber.java.After;
import io.cucumber.java.Before;
import io.cucumber.java.Scenario;
import utils.Tools;

public class BeforeAfter extends Tools {

      @Before
      public void setup(Scenario scenario) {
             Driver.init();
      }
```

```
		@After
		public void tearDown() {
			driver.quit();
		}
	}
```

Driver.java

```
	package com.Swiggy.TestSteps;

	import org.openqa.selenium.edge.EdgeDriver;

	import com.Swiggy.Pages.CheckoutPage;
	import com.Swiggy.Pages.DishesAndCartPage;
	import com.Swiggy.Pages.LandingPage;
	import com.Swiggy.Pages.SearchResultsRestaurant;

	import utils.Tools;

	public class Driver extends Tools {
		protected static LandingPage landingPage;
		protected static SearchResultsRestaurant searchResultsRestaurant;
		protected static DishesAndCartPage dishesAndCartPage;
		protected static CheckoutPage checkoutPage;

		public static void init() {
			driver = new EdgeDriver();
			driver.manage().window().maximize();
			driver.get("https://swiggy.com");
			landingPage = new LandingPage(driver);
			searchResultsRestaurant = new SearchResultsRestaurant(driver);
			dishesAndCartPage = new DishesAndCartPage(driver);
			checkoutPage = new CheckoutPage(driver);
		}

	}
```

TestRunner.java

```
	package com.Swiggy.TestSteps;

	import io.cucumber.testng.AbstractTestNGCucumberTests;
	import io.cucumber.testng.CucumberOptions;

	@CucumberOptions(features = "Features", glue = "com.Swiggy.TestSteps")

	public class TestRunner extends AbstractTestNGCucumberTests {

	}
```

VerifyTextSteps.java

```
package com.Swiggy.TestSteps;

import org.testng.Assert;

import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;

public class VerifyTextSteps extends Driver {

	@Given("a user is on the landing page")
	public void a_user_is_on_the_landing_page() {

	}

	@When("he type the location and clicks 1st location")
	public void he_type_the_location_and_clicks_1st_location() {
		LandingPage.enterlLocation("Hyderabad");
	}

	@When("he click 1st restaurant under Top restaurant chains in
Hyderabad")
	public void
he_click_1st_restaurant_under_top_restaurant_chains_in_hyderabad() {
		searchResultsRestaurant.clickFirstRestaurant();
	}

	@When("he clicks add button on first listed dish")
	public void he_clicks_add_button_on_first_listed_dish() {
		dishesAndCartPage.clickFirstRestaurant();
	}

	@When("hover over the cart icon in right top corner")
	public void hover_over_the_cart_icon_in_right_top_corner() throws
InterruptedException {
		dishesAndCartPage.hoverOverCartIcon();
	}

	@When("he clicks Checkout button")
	public void he_clicks_checkout_button() {
		dishesAndCartPage.clickCheckout();
	}

	@Then("verify the text {string}")
	public void verify_the_text(String string) {
		String expectedText = string;
		String actualText = checkoutPage.verifyText();
		Assert.assertEquals(actualText, expectedText);
	}

}
```

Tools.java

```java
package utils;

import org.openqa.selenium.WebDriver;

public class Tools {
	protected static WebDriver driver;
}
```

pom.xml

```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.Swiggy</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>7.1.0</version>
		</dependency>
		<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-
java -->
		<dependency>
			<groupId>io.cucumber</groupId>
			<artifactId>cucumber-java</artifactId>
			<version>7.1.0</version>
		</dependency>
		<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-
testng -->
		<dependency>
			<groupId>io.cucumber</groupId>
			<artifactId>cucumber-testng</artifactId>
			<version>7.1.0</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/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>4.10.0</version>
		</dependency>
		<dependency>
			<groupId>com.google.guava</groupId>
			<artifactId>guava</artifactId>
			<version>31.0.1-jre</version>
		</dependency>
		<dependency>
			<groupId>com.aventstack</groupId>
			<artifactId>extentreports</artifactId>
			<version>3.1.5</version>
		</dependency>
		<dependency>
			<groupId>com.sun.mail</groupId>
			<artifactId>javax.mail</artifactId>
			<version>1.6.2</version>
		</dependency>
```

```
                <dependency>
                        <groupId>javax.activation</groupId>
                        <artifactId>activation</artifactId>
                        <version>1.1</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>
```

```xml
        </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>
```

testng.xml

```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.TestSteps.TestRunner"></class>
		</classes>
	</test> <!--Test -->
</suite> <!--Suite -->
```