**Pizzahut.feature**

@Smoke

Feature: Pizzahut Automation

Scenario Outline: Validate Pizzahut pizza order flow

Given User launch Pizzahut application with "<URL>"

When User wait for auto location black pop up screen

Then User close the pop up screen

And User see pop up for delivery asking for enter location

Then User type address as "<Location>"

And User select first auto populate drop down option

When User navigate to deails page

Then User validate vegetarian radio button flag is off

And User clicks on Pizzas menu bar option

When User select add button of any pizza from Recommended

Then User see that the pizza is getting added under Your Basket

And User validate pizza price plus Tax is checkout price

Then User validate checkout button contains Item count

And User validate checkout button contains total price count

Then User clicks on Drinks option

And User select Pepsi option to add into the Basket

Then User see 2 items are showing under checkout button

And User see total price is now more than before

Then User remove the Pizza item from Basket

And see Price tag got removed from the checkout button

And User see 1 item showing in checkout button

Then User Clicks on Checkout button

And User see minimum order required pop up is getting displayed

Examples:

|URL | Location|

|https://www.pizzahut.co.in/|kharagpur|

**StepDefinitions.java**

package stepDefinitions;

import static org.junit.Assert.assertEquals;

import static org.junit.Assert.assertNotEquals;

import static org.junit.Assert.assertTrue;

import static org.testng.Assert.assertFalse;

import java.time.Duration;

import java.util.NoSuchElementException;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

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

import com.aventstack.extentreports.ExtentReports;

import com.aventstack.extentreports.ExtentTest;

import com.aventstack.extentreports.Status;

import io.cucumber.java.After;

import io.cucumber.java.Before;

import io.cucumber.java.en.And;

import io.cucumber.java.en.Given;

import io.cucumber.java.en.Then;

import io.cucumber.java.en.When;

import io.github.bonigarcia.wdm.WebDriverManager;

import main.java.utils.ExtentReportManager;

import pizzahutPageObjects.PizzaHutPageObjects;

public class StepDefinitions {

private ExtentReports extent;

ExtentTest test = ExtentReportManager.createTest("Pizzahut Automation Test");

public static WebDriver driver;

WebDriverWait wait;

PizzaHutPageObjects pizzaHutPage;

String pizzaPrice, restaurantCharges, taxAmount, totalPriceBefore, totalPriceAfter, totalPriceAfter1;

@Before

public void setUpExtentReport(io.cucumber.java.Scenario scenario) {

extent = ExtentReportManager.getInstance();

test = extent.createTest(scenario.getName());

}

@After

public void tearDownExtentReport(io.cucumber.java.Scenario scenario){

if (scenario.isFailed()) {

test.log(Status.FAIL, "Scenario failed: " + scenario.getName());

} else {

test.log(Status.PASS, "Scenario passed: " + scenario.getName());

}

if(driver != null) {

driver.quit();

}

if(extent != null) {

extent.flush();

}

}

{

/\* Setting up the project \*/}

@Given("User launch Pizzahut application with {string}")

public void user\_launch\_pizzahut\_application\_with(String url) {

driver = new ChromeDriver();

WebDriverManager.chromedriver().clearDriverCache().setup();

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

pizzaHutPage = new PizzaHutPageObjects(driver);

try {

driver.get(url);

test.log(Status.PASS, "Launched Application with URL: " + url);

} catch (Exception e) {

test.log(Status.FAIL, "Failed to launch application with URL: " + url);

throw e;

}

}

@When("User wait for auto location black pop up screen")

public void user\_wait\_for\_auto\_location\_black\_pop\_up\_screen() throws InterruptedException {

try {

pizzaHutPage.waitForAutoLocationBlackPopUp();

test.log(Status.PASS, "Auto location black pop up is displayed");

} catch(NoSuchElementException e) {

test.log(Status.INFO, "Location pop up did not appear, continuing");

}catch (Exception e) {

System.out.println(e);

test.log(Status.INFO, "Unexpected error: "+ e.getMessage());

}

}

@Then("User close the pop up screen")

public void user\_close\_the\_pop\_up\_screen() throws InterruptedException {

try {

pizzaHutPage.closePopUp();

test.log(Status.PASS, "Closed the pop up screen");

}catch(NoSuchElementException e) {

test.log(Status.INFO, "Pop up screen was not found to close, continuing");

}catch (Exception e) {

System.out.println(e);

test.log(Status.INFO, "Failed to close the pop up screen: " + e.getMessage());

}

}

@And("User see pop up for delivery asking for enter location")

public void user\_see\_pop\_up\_for\_delivery\_asking\_for\_enter\_location() throws InterruptedException {

wait = new WebDriverWait(driver, Duration.ofSeconds(10));

try {

pizzaHutPage.waitForLocationInput();

test.log(Status.PASS, "Location input field is displayed");

} catch (Exception e) {

System.out.println(e);

test.log(Status.FAIL, "Location input field is not displayed");

throw e;

}

}

WebElement locationInput;

@Then("User type address as {string}")

public void user\_type\_address\_as(String location) throws InterruptedException {

try {

pizzaHutPage.typeAddress(location);

test.log(Status.PASS, "Typed address: " + location);

} catch (Exception e) {

System.out.println(e);

test.log(Status.FAIL, "Failed to type address: " + location);

throw e;

}

}

@And("User select first auto populate drop down option")

public void user\_select\_first\_auto\_populate\_drop\_down\_option() {

try {

pizzaHutPage.selectFirstAutoPopulateDropdownOption();

test.log(Status.PASS, "Selected first auto populate drop down option");

} catch (Exception e) {

System.out.println(e);

test.log(Status.FAIL, "Failed to select first auto populate drop down option");

throw e;

}

}

@When("User navigate to deails page")

public void user\_navigate\_to\_deails\_page() throws InterruptedException {

try {

pizzaHutPage.waitForDealsPage();

test.log(Status.PASS, "Deals page is displayed");

} catch (Exception e) {

System.out.println(e);

test.log(Status.FAIL, "Deals page is not displayed");

throw e;

}

}

@Then("User validate vegetarian radio button flag is off")

public void user\_validate\_vegetarian\_radio\_button\_flag\_is\_off() throws InterruptedException {

try {

assertFalse(pizzaHutPage.vegetarianRadioBtnFlag.isSelected());

if (!pizzaHutPage.vegetarianRadioBtnFlag.isSelected()) {

System.out.println("radio button is off");

} else {

System.out.println("radio button is enabled");

}

test.log(Status.PASS, "Vegetarian radio button flag is off");

} catch (Exception e) {

System.out.println(e);

test.log(Status.FAIL, "Vegetarian radio button flag is not off");

throw e;

}

}

@And("User clicks on Pizzas menu bar option")

public void user\_clicks\_on\_pizzas\_menu\_bar\_option() {

try {

pizzaHutPage.clickPizzasMenuBarOption();

System.out.println(pizzaHutPage.pizzasMenuBarOption.getText());

test.log(Status.PASS, "Clicked on Pizzas menu bar option");

} catch (Exception e) {

System.out.println(e);

test.log(Status.FAIL, "Failed to click on Pizzas menu bar option");

throw e;

}

}

@When("User select add button of any pizza from Recommended")

public void user\_select\_add\_button\_of\_any\_pizza\_from\_recommended() {

String targetPizzaName = "Southern Fiery Chicken";

try {

pizzaHutPage.selectAddButtonOfAnyPizzaFromRecommended(targetPizzaName);

test.log(Status.PASS, "Selected pizza: " + targetPizzaName);

} catch (Exception e) {

System.out.println(e);

test.log(Status.FAIL, "Failed to select pizza: " + targetPizzaName);

throw e;

}

}

@Then("User see that the pizza is getting added under Your Basket")

public void user\_see\_that\_the\_pizza\_is\_getting\_added\_under\_your\_basket() {

try {

pizzaHutPage.getBasketItemProductTitle();

System.out.println(pizzaHutPage.basketItemProductTitle.getText());

test.log(Status.PASS, "Pizza is added to Your Basket: " + pizzaHutPage.basketItemProductTitle.getText());

} catch (Exception e) {

System.out.println(e);

test.log(Status.FAIL, "Pizza is not added to Your Basket");

throw e;

}

}

@And("User validate pizza price plus Tax is checkout price")

public void user\_validate\_pizza\_price\_plus\_tax\_is\_checkout\_price() {

try {

pizzaPrice = pizzaHutPage.getPizzaPrice();

restaurantCharges = pizzaHutPage.getRestaurantCharges();

taxAmount = pizzaHutPage.getTaxAmount();

totalPriceBefore = pizzaHutPage.getTotalPriceBefore();

System.out.println("PizzaPrice: " + pizzaPrice);

System.out.println("Restaurant Handling Charges: " + restaurantCharges);

System.out.println("taxAmount: " + taxAmount);

System.out.println("totalPayment: " + totalPriceBefore);

String cleanedPizzaPrice = pizzaPrice.replaceAll("[^\\d.]", "").trim();

String cleanedRestaurantCharges = restaurantCharges.replaceAll("[^\\d.]", "").trim();

String cleanedTaxAmount = taxAmount.replaceAll("[^\\d.]", "").trim();

String cleanedTotalPriceBefore = totalPriceBefore.replaceAll("[^\\d.]", "").trim();

double price = Double.parseDouble(cleanedPizzaPrice);

double restCharge = Double.parseDouble(cleanedRestaurantCharges);

double tax = Double.parseDouble(cleanedTaxAmount);

double total = Double.parseDouble(cleanedTotalPriceBefore);

System.out.println("Price + Tax should match Total: " + (price + restCharge + tax));

assertEquals("Price + Restaurant Charges + Tax should match Total", price + restCharge + tax, total, 0.01);

System.out.println(

"Price: " + price + ", Restaurant Charges: " + restCharge + ", Tax: " + tax + ", total: " + total);

test.log(Status.PASS, "Pizza price plus restaurant charges and tax matches the total price");

} catch (Exception e) {

System.out.println(e);

test.log(Status.FAIL, "Pizza price plus restaurant charges and tax does not match the total price");

throw e;

}

}

@Then("User validate checkout button contains Item count")

public void user\_validate\_checkout\_button\_contains\_item\_count() {

try {

String itemText = pizzaHutPage.getCheckoutBtnItemText();

assertTrue(itemText.contains("1 item"));

System.out.println("item count text - passed!");

test.log(Status.PASS, "Checkout button contains item count: " + itemText);

} catch (Exception e) {

System.out.println(e);

test.log(Status.FAIL, "Checkout button does not contain item count");

throw e;

}

}

@And("User validate checkout button contains total price count")

public void user\_validate\_checkout\_button\_contains\_total\_price\_count() {

try {

String totalPriceCount = pizzaHutPage.getTotalPriceCount();

System.out.println("Total Price: " + totalPriceCount);

assertTrue(totalPriceCount.contains(totalPriceBefore));

test.log(Status.PASS, "Checkout button contains total price count: " + totalPriceCount);

} catch (Exception e) {

System.out.println(e);

test.log(Status.FAIL, "Checkout button does not contain total price count");

throw e;

}

}

@Then("User clicks on Drinks option")

public void user\_clicks\_on\_drinks\_option() {

try {

pizzaHutPage.clickDrinksOption();

test.log(Status.PASS, "Clicked on Drinks option");

} catch (Exception e) {

System.out.println(e);

test.log(Status.FAIL, "Failed to click on Drinks option");

throw e;

}

}

@And("User select Pepsi option to add into the Basket")

public void user\_select\_pepsi\_option\_to\_add\_into\_the\_basket() {

String targetDrinkName = "Pepsi";

try {

pizzaHutPage.selectPepsiOptionToAddIntoBasket(targetDrinkName);

test.log(Status.PASS, "Selected drink: " + targetDrinkName);

} catch (Exception e) {

System.out.println(e);

test.log(Status.FAIL, "Failed to select drink: " + targetDrinkName);

throw e;

}

}

@Then("User see {int} items are showing under checkout button")

public void user\_see\_items\_are\_showing\_under\_checkout\_button(Integer int1) {

try {

assertTrue(pizzaHutPage.checkoutBtnItemText.getText().contains(int1 + " items"));

System.out.println(pizzaHutPage.checkoutBtnItemText.getText());

test.log(Status.PASS, "Checkout button shows " + int1 + " items");

} catch (Exception e) {

System.out.println(e);

test.log(Status.FAIL, "Checkout button does not show " + int1 + " items");

throw e;

}

}

@And("User see total price is now more than before")

public void user\_see\_total\_price\_is\_now\_more\_than\_before() {

try {

totalPriceAfter = pizzaHutPage.getTotalPriceBefore();

assertNotEquals(totalPriceBefore, totalPriceAfter);

System.out.println("total before: " + totalPriceBefore + "total after: " + totalPriceAfter);

System.out.println("total price is now more than before!");

test.log(Status.PASS, "Total price is now more than before: " + totalPriceAfter);

} catch (Exception e) {

System.out.println(e);

test.log(Status.FAIL, "Total price is not more than before");

throw e;

}

}

@Then("User remove the Pizza item from Basket")

public void user\_remove\_the\_pizza\_item\_from\_basket() {

try {

pizzaHutPage.removePizzaFromBasket();

test.log(Status.PASS, "Removed pizza from basket");

} catch (Exception e) {

System.out.println(e);

test.log(Status.FAIL, "Failed to remove pizza from basket");

throw e;

}

}

@And("see Price tag got removed from the checkout button")

public void see\_price\_tag\_got\_removed\_from\_the\_checkout\_button() {

try {

pizzaHutPage.waitForPriceTagRemoved();

assertFalse(pizzaHutPage.priceTagRemoved.getText().contains(totalPriceBefore));

test.log(Status.PASS, "Price tag got removed from the checkout button");

} catch (Exception e) {

System.out.println(e);

test.log(Status.FAIL, "Price tag did not get removed from the checkout button");

throw e;

}

}

@And("User see {int} item showing in checkout button")

public void user\_see\_item\_showing\_in\_checkout\_button(Integer int1) {

try {

assertTrue(pizzaHutPage.checkoutBtnItemText.getText().contains(int1 + " item"));

System.out.println(pizzaHutPage.checkoutBtnItemText.getText());

test.log(Status.PASS, "Checkout button shows " + int1 + " item");

} catch (Exception e) {

System.out.println(e);

test.log(Status.FAIL, "Checkout button does not show " + int1 + " item");

throw e;

}

}

@Then("User Clicks on Checkout button")

public void user\_clicks\_on\_checkout\_button() {

try {

pizzaHutPage.clickCheckoutButton();

test.log(Status.PASS, "Clicked on Checkout button");

} catch (Exception e) {

System.out.println(e);

test.log(Status.FAIL, "Failed to click on Checkout button");

throw e;

}

}

@And("User see minimum order required pop up is getting displayed")

public void user\_see\_minimum\_order\_required\_pop\_up\_is\_getting\_displayed() {

try {

assertTrue(pizzaHutPage.isMinOrderPopupDisplayed());

System.out.println(pizzaHutPage.minOrderPopup.isDisplayed());

test.log(Status.PASS, "Minimum order required popup is displayed");

} catch (Exception e) {

System.out.println(e);

test.log(Status.FAIL, "Minimum order required popup is not displayed");

throw e;

}

}

}

**PizzaHutPageObjects.java**

package pizzahutPageObjects;

import java.time.Duration;

import java.util.List;

import org.openqa.selenium.By;

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 PizzaHutPageObjects {

WebDriver driver;

WebDriverWait wait;

// Pop up black screen

@FindBy(xpath = "//div[@class='pl-10 pr-10']")

public WebElement autoLocationBlackPopUp;

// Pop up close button

@FindBy(xpath = "//button[@class='icon-close--white p-30 absolute top-0 right-0 mr-10 mt-10']")

public WebElement locationPopUpClsbtn;

// Location input

@FindBy(xpath = "//input[contains(@class,'search--hut')]")

public WebElement locationInput;

// First auto-populate dropdown option

@FindBy(xpath = "//div[@class='pt-5 border-t overflow-scrolling-touch']")

public List<WebElement> locationDropdownOptions;

@FindBy(xpath = "(//button[starts-with(@id, 'PlacesAutocomplete')])[1]")

public WebElement firstAutoPopulateOption;

@FindBy(xpath = "//button[contains(@class,'text-center')]")

public WebElement startOrderTimeBtn;

// Deals page navigation

@FindBy(xpath = "(//a[@href='/order/deals/'])[2]")

public WebElement dealsPageLink;

// Vegetarian radio button

@FindBy(xpath = "(//span[contains(@class, 'py-4 px-5 border rounded-full flex items-center')])[1]")

public WebElement vegetarianRadioBtnFlag;

// Pizzas menu bar option

@FindBy(xpath = "(//a[@href='/order/pizzas/'])[2]")

public WebElement pizzasMenuBarOption;

// Pizza containers (Recommended)

@FindBy(css = ".sc-fznXWL.sc-fznXWL.product-grid")

public List<WebElement> pizzaContainers;

// Add button inside pizza (relative, so handled in method)

// Basket

@FindBy(id = "basket")

public WebElement yourBasket;

@FindBy(xpath = "//div[contains(@class, 'basket-item-product-title')]")

public WebElement basketItemProductTitle;

@FindBy(xpath = "//div[contains(@class,'basket-item-product-price')]")

public WebElement basketItemProductPrice;

@FindBy(xpath = "//div[@class='text-12']/div[contains(@class,'supplement-value')]")

public WebElement restaurantCharges;

@FindBy(xpath = "(//div[contains(@class,'items-start')]/span)[4]")

public WebElement taxAmount;

@FindBy(xpath = "//a/span/span[contains(@data-synth,'basket-value')]")

public WebElement basketValue;

@FindBy(xpath = "//span/span[contains(@class,'bg-green-dark ')]")

public WebElement checkoutBtnItemText;

@FindBy(xpath = "//a[@href='/order/checkout/']/span[3]/span")

public WebElement totalPriceCount;

// Drinks option

@FindBy(xpath = "(//a[@href='/order/drinks/'])[2]")

public WebElement drinksOption;

// Drink containers

@FindBy(xpath = "//div[contains(@class,'product-grid')]")

public List<WebElement> drinkContainers;

// Add button for drink (handled in method)

@FindBy(xpath = "//button[contains(@class,'button--green')]/span/span[text()='Add']")

public WebElement addDrinkBtn;

// Remove pizza from basket

@FindBy(xpath = "(//button[contains(@class,'icon-close')])[1]")

public WebElement removePizzaBtn;

// Price tag removed from checkout

@FindBy(xpath = "(//span[contains(@class,'absolute')]/span)[2]")

public WebElement priceTagRemoved;

// Checkout button

@FindBy(xpath = "(//span[contains(@class,'absolute')]/span)[2]")

public WebElement checkoutButton;

// Minimum order required popup

@FindBy(xpath = "//div[contains(@class,'pt-20 bg-white')]")

public WebElement minOrderPopup;

public PizzaHutPageObjects(WebDriver driver) {

this.driver = driver;

this.wait = new WebDriverWait(driver, Duration.ofSeconds(10));

PageFactory.initElements(driver, this);

}

// Wait for auto location black pop up

public void waitForAutoLocationBlackPopUp() {

wait.until(ExpectedConditions.visibilityOf(autoLocationBlackPopUp));

}

// Close pop up

public void closePopUp() {

locationPopUpClsbtn.click();

}

// Wait for location input

public void waitForLocationInput() {

wait.until(ExpectedConditions.visibilityOf(locationInput));

}

// Type address

public void typeAddress(String address) {

locationInput.sendKeys(address);

}

// Select first auto populate dropdown option

public void selectFirstAutoPopulateDropdownOption() {

wait.until(ExpectedConditions

.presenceOfElementLocated(By.xpath("(//button[starts-with(@id, 'PlacesAutocomplete')])[1]")));

wait.until(ExpectedConditions.elementToBeClickable(firstAutoPopulateOption));

if (!locationDropdownOptions.isEmpty()) {

locationDropdownOptions.get(0).click();

} else {

throw new RuntimeException("No options found");

}

try {

wait.until(

ExpectedConditions.presenceOfElementLocated(By.xpath("//button[contains(@class,'text-center')]")));

startOrderTimeBtn.click();

} catch (Exception e) {

// ignore

}

}

// Wait for deals page

public void waitForDealsPage() {

wait.until(ExpectedConditions.visibilityOf(dealsPageLink));

}

// Validate vegetarian radio button flag is off

public boolean isVegetarianRadioBtnFlagOff() {

return !vegetarianRadioBtnFlag.isSelected();

}

// Click pizzas menu bar option

public void clickPizzasMenuBarOption() {

pizzasMenuBarOption.click();

}

// Select add button of any pizza from Recommended

public void selectAddButtonOfAnyPizzaFromRecommended(String targetPizzaName) {

wait.until(

ExpectedConditions.presenceOfAllElementsLocatedBy(By.cssSelector(".sc-fznXWL.sc-fznXWL.product-grid")));

for (WebElement pizza : pizzaContainers) {

if (pizza.getText().trim().contains(targetPizzaName)) {

WebElement addBtn = pizza.findElement(By.xpath(".//button/span/span[text()='Add']"));

wait.until(ExpectedConditions.elementToBeClickable(addBtn)).click();

return;

}

}

throw new RuntimeException("No matching pizza found");

}

// Get basket item product title

public String getBasketItemProductTitle() {

wait.until(ExpectedConditions

.presenceOfElementLocated(By.xpath("//div[contains(@class, 'basket-item-product-title')]")));

return basketItemProductTitle.getText();

}

// Get pizza price, restaurant charges, tax, total

public String getPizzaPrice() {

return basketItemProductPrice.getText();

}

public String getRestaurantCharges() {

return restaurantCharges.getText();

}

public String getTaxAmount() {

return taxAmount.getText();

}

public String getTotalPriceBefore() {

return basketValue.getText();

}

// Get checkout button item count

public String getCheckoutBtnItemText() {

return checkoutBtnItemText.getText();

}

// Get total price count

public String getTotalPriceCount() {

return totalPriceCount.getText();

}

// Click drinks option

public void clickDrinksOption() {

drinksOption.click();

}

// Select Pepsi option to add into the Basket

public void selectPepsiOptionToAddIntoBasket(String targetDrinkName) {

wait.until(

ExpectedConditions.presenceOfAllElementsLocatedBy(By.xpath("//div[contains(@class,'product-grid')]")));

for (WebElement drink : drinkContainers) {

if (drink.getText().contains(targetDrinkName)) {

wait.until(ExpectedConditions.elementToBeClickable(addDrinkBtn)).click();

return;

}

}

throw new RuntimeException("No drink found");

}

// Remove pizza from basket

public void removePizzaFromBasket() {

removePizzaBtn.click();

}

// Wait for price tag to be removed from checkout button

public void waitForPriceTagRemoved() {

wait.until(ExpectedConditions

.invisibilityOfElementLocated(By.xpath("//a/span/span[contains(@data-synth,'basket-value')]")));

wait.until(

ExpectedConditions.presenceOfElementLocated(By.xpath("(//span[contains(@class,'absolute')]/span)[2]")));

}

// Get price tag removed text

public String getPriceTagRemovedText() {

return priceTagRemoved.getText();

}

// Click checkout button

public void clickCheckoutButton() {

checkoutButton.click();

}

// Wait for minimum order required popup

public boolean isMinOrderPopupDisplayed() {

wait.until(ExpectedConditions.visibilityOf(minOrderPopup));

return minOrderPopup.isDisplayed();

}

}

**TestRunner.java**

package testRunner;

import org.junit.runner.RunWith;

import io.cucumber.junit.Cucumber;

import io.cucumber.junit.CucumberOptions;

@RunWith(Cucumber.class)

@CucumberOptions(

features = "src/test/pizzahut.feature",

glue= {"stepDefinitions"},

tags= "@Smoke",

plugin = {"pretty", "html:target/cucumber/cucumber-reports" }

)

public class TestRunner {

}

**ExtentReportManager.java**

package main.java.utils;

import com.aventstack.extentreports.ExtentReports;

import com.aventstack.extentreports.ExtentTest;

import com.aventstack.extentreports.reporter.ExtentHtmlReporter;

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

public class ExtentReportManager {

private static ExtentReports extent;

private static ExtentTest test;

public static ExtentReports getInstance() {

if(extent == null) {

ExtentHtmlReporter htmlReporter = new ExtentHtmlReporter("test-output/ExtentReport.html");

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

htmlReporter.config().setDocumentTitle("Pizzahut Automation Report");

htmlReporter.config().setReportName("Automation Test Results");

extent = new ExtentReports();

extent.attachReporter(htmlReporter);

}

return extent;

}

public static ExtentTest createTest(String testName){

test = getInstance().createTest(testName);

return test;

}

public static ExtentTest getTest(){

return test;

}

}

**Pom.xml**

<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>PizzahutAutomationWithCucumber</groupId>

<artifactId>PizzahutAutomationWithCucumber</artifactId>

<version>0.0.1-SNAPSHOT</version>

<build>

<sourceDirectory>src</sourceDirectory>

<plugins>

<plugin>

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

<version>3.13.0</version>

<configuration>

<release>11</release>

</configuration>

</plugin>

</plugins>

</build>

<dependencies>

<!-- 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>

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

<dependency>

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

<artifactId>selenium-java</artifactId>

<version>4.32.0</version>

</dependency>

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

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>7.11.0</version>

</dependency>

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

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-junit</artifactId>

<version>7.22.1</version>

</dependency>

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

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-java</artifactId>

<version>7.22.1</version>

</dependency>

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

<dependency>

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

<artifactId>webdrivermanager</artifactId>

<version>6.1.0</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.poi/poi-ooxml -->

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi-ooxml</artifactId>

<version>5.4.1</version>

</dependency>

<!-- https://mvnrepository.com/artifact/com.aventstack/extentreports -->

<dependency>

<groupId>com.aventstack</groupId>

<artifactId>extentreports</artifactId>

<version>5.1.2</version>

</dependency>

</dependencies>

</project>

**ExtentReport.html**

