**Flight Booking**

**flightbooking.feature**

**Path: /FlightBooking2/src/test/java/com/flightbooking/feature/flightbooking.feature**

Feature: Flight booking

Scenario Outline: Select the best itinerary based on fastest and cheapest travel

Given I open cleartrip website

When I Search for "<source>" and "<destination>"

Then Select fastest and cheapest travel itinerary

Examples:

| source | destination |

| HYD | MAA |

**StepDefinition file: Flightbooking.java**

**Path:**

**/FlightBooking2/src/test/java/com/flightbooking/stepdefinition/Flightbooking.java**

package com.flightbooking.stepdefinition;

import com.flightbooking.Pages.HomePage;

import cucumber.api.java.en.Given;

import cucumber.api.java.en.Then;

import cucumber.api.java.en.When;

public class Flightbooking {

@Given("^I open cleartrip website$")

public void i\_open\_cleartrip\_website() throws Throwable {

HomePage.launchurl();

}

@When("^I Search for \"([^\"]\*)\" and \"([^\"]\*)\"$")

public void i\_Search\_for\_source\_and\_destination(String source, String destination) throws Throwable {

HomePage.enterSource(source);

HomePage.enterDestination(destination);

HomePage.enterDate();

HomePage.searchFlights();

}

@Then("^Select fastest and cheapest travel itinerary$")

public void getbestTravel() throws InterruptedException {

HomePage.getFastestAndCheapestItinerary();

}

}

**HomePage.java**

**Path:** **/FlightBooking2/src/test/java/com/flightbooking/Pages/HomePage.java**

package com.flightbooking.Pages;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

import org.openqa.selenium.By;

import org.openqa.selenium.JavascriptExecutor;

import org.openqa.selenium.Keys;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.support.FindBy;

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

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

import org.testng.Assert;

import com.flightbooking.runner.TestRunner;

import com.flightbooking.utility.Log;

public class HomePage extends TestRunner {

static WebDriverWait wait=new WebDriverWait(driver, 60);;

static List<WebElement> fares;

static List<WebElement> allDuration;

static ArrayList<Integer> durationInSeconds;

@FindBy(xpath="//a[@class='ctBrand']/span[@title='Cleartrip ']")

private static WebElement title;

@FindBy(id="FromTag")

private static WebElement departFrom;

@FindBy(id="ToTag")

private static WebElement goingTo;

@FindBy(id="DepartDate")

private static WebElement departDate;

@FindBy(id="SearchBtn")

private static WebElement searchFlights;

@FindBy(xpath="//div[@class='col-5']/div/aside/div[4]/div[3]/div[2]/div[1]/label[4]/div[1]/span")

private static WebElement eveningCheckbox;

public static void launchurl()

{

String expectedUrl="https://www.cleartrip.com/";

driver.get(expectedUrl);

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

String actualUrl=driver.getCurrentUrl();

Assert.assertEquals(expectedUrl, actualUrl);

}

public static void enterSource(String source)

{

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

departFrom.clear();

departFrom.sendKeys(source);

}

public static void enterDestination(String destination)

{

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

goingTo.clear();

goingTo.sendKeys(destination);

}

public static void enterDate()

{

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

departDate.clear();

departDate.sendKeys("Sat, 15 Aug, 2020");

departDate.sendKeys(Keys.ENTER);

}

public static void searchFlights() throws InterruptedException

{

JavascriptExecutor js=(JavascriptExecutor)driver;

js.executeScript("arguments[0].scrollIntoView(true);", searchFlights);

searchFlights.click();

Thread.sleep(10000);

}

public static ArrayList<Integer> getFares() throws InterruptedException

{

Thread.sleep(5000);

ArrayList<Integer> pricesList=new ArrayList<Integer>();

fares=driver.findElements(By.xpath("//th[@id='BaggageBundlingTemplate']"));

if(fares.isEmpty()||fares==null)

{

Thread.sleep(10000);

fares=driver.findElements(By.xpath("//div[@data-ct-handle='solutionPrice']/p"));

}

for(int i=0;i<fares.size();i++)

{

int fare=Integer.parseInt(fares.get(i).getText().replaceAll("[^0-9]", ""));

pricesList.add(fare);

}

for(int flightfares:pricesList)

{

Log.info("flightfares"+flightfares);

}

return pricesList;

}

public static void getFastestAndCheapestItinerary() throws InterruptedException {

Thread.sleep(3000);

List<WebElement> allBookbtn;

ArrayList<Integer> alltimes=getDuration();

int minTime=Collections.min(alltimes);

ArrayList<Integer> allPrices=getFares();

int minPrice=Collections.min(allPrices);

allBookbtn = driver.findElements(By.xpath("//div[@class='flex flex-right nmt-1 ms-grid-column-4']/button"));

if(allBookbtn.isEmpty()||allBookbtn==null)

{

allBookbtn=driver.findElements(By.xpath("//td[@class='price actionPrice ']/button"));

}

Thread.sleep(3000);

for(int k=0;k<durationInSeconds.size();k++){

if(durationInSeconds.get(k)==minTime){

for(int i=0;i<fares.size();i++){

Integer priceInt1 = Integer.valueOf(fares.get(i).getText().replaceAll("[^0-9]", ""));

Log.info("Price in Integer format"+priceInt1);

if(priceInt1==minPrice){

allBookbtn.get(i).click();

break;

}

}

}

}

Thread.sleep(6000);

driver.findElement(By.cssSelector("body")).sendKeys(Keys.CONTROL+"\t");

Thread.sleep(6000);

}

public static ArrayList<Integer> getDuration() throws InterruptedException

{

String[] lst;

Thread.sleep(5000);

allDuration = driver.findElements(By.xpath("//div[@class='ms-grid-column-3 ms-grid-row-1']/p"));

if(allDuration.isEmpty()||allDuration==null)

{

allDuration=driver.findElements(By.xpath("//th[@class='duration']"));

}

durationInSeconds=new ArrayList<Integer>();

for(int j=0;j<allDuration.size();j++)

{

String duration=allDuration.get(j).getText();

if(duration.contains("h")){

lst=duration.split("\\s+");

Log.info(lst[0]);

Log.info(lst[1]);

int hours=Integer.parseInt(lst[0].replaceAll("[^0-9]", ""));

int minutes=Integer.parseInt(lst[1].replaceAll("[^0-9]", ""));

int seconds=(hours\*60\*60)+(minutes\*60);

durationInSeconds.add(seconds);

}

else

{

int minutes=Integer.parseInt(duration.replaceAll("[^0-9]", ""));

int seconds=minutes\*60;

durationInSeconds.add(seconds);

}

}

for(int timeDuration:durationInSeconds)

{

Log.info("Time duration in seconds"+timeDuration);

}

return durationInSeconds;

}

}

**TestRunner.java**

**Path:** **/FlightBooking2/src/test/java/com/flightbooking/runner/TestRunner.java**

**package** com.flightbooking.runner;

**import** org.apache.log4j.xml.DOMConfigurator;

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

**import** org.openqa.selenium.chrome.ChromeDriver;

**import** org.openqa.selenium.chrome.ChromeOptions;

**import** org.openqa.selenium.safari.SafariDriver;

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

**import** org.testng.annotations.AfterClass;

**import** org.testng.annotations.BeforeClass;

**import** org.testng.annotations.DataProvider;

**import** org.testng.annotations.Parameters;

**import** org.testng.annotations.Test;

**import** com.flightbooking.Pages.HomePage;

**import** cucumber.api.CucumberOptions;

**import** cucumber.api.testng.CucumberFeatureWrapper;

**import** cucumber.api.testng.TestNGCucumberRunner;

@CucumberOptions(features = "src/test/java/com/flightbooking/feature",glue={"com.flightbooking.stepdefinition"},plugin = {"html:target/cucumber-html-report","json:target/cucumber.json","pretty:target/cucumber-pretty.txt","junit:target/cucumber-results.xml" })

//public class TestRunner extends AbstractTestNGCucumberTests{

**public** **class** TestRunner{

**public** **static** WebDriver *driver*;

**private** TestNGCucumberRunner testNGCucumberRunner;

@BeforeClass(alwaysRun = **true**)

**public** **void** setUpClass() **throws** Exception {

testNGCucumberRunner = **new** TestNGCucumberRunner(**this**.getClass());

}

@Test(groups = "cucumber", description = "Runs Cucumber Feature", dataProvider = "features")

**public** **void** feature(CucumberFeatureWrapper cucumberFeature) {

testNGCucumberRunner.runCucumber(cucumberFeature.getCucumberFeature());

}

@DataProvider

**public** Object[][] features(){

**return** testNGCucumberRunner.provideFeatures();

}

@AfterClass(alwaysRun = **true**)

**public** **void** tearDownCLass()

{

testNGCucumberRunner.finish();

}

@BeforeClass

@Parameters("browser")

**public** WebDriver launchChromeBrowser(String browser) {

DOMConfigurator.*configure*("src/test/java/com/flightbooking/utility/log4j.xml");

**if** (browser.equals("chrome")) {

ChromeOptions options=**new** ChromeOptions();

options.addArguments("--disable-notifications");

System.*setProperty*("webdriver.chrome.driver", "src/test/resources/chromedriver.exe");

*driver* = **new** ChromeDriver(options);

}

**if** (browser.equals("safari")) {

*driver* = **new** SafariDriver();

}

PageFactory.*initElements*(TestRunner.*driver*, HomePage.**class**);

**return** *driver*;

}

@AfterClass(alwaysRun = **true**)

**public** **void** closeBrowser()

{

*driver*.close();

}

}

**TestNG.xml**

**Path: E:\AutomationChallenge\FlightBooking2\src\test\resources\TestNG\testng.xml**

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

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

<suite name=*"FlightBooking"*>

<test name=*"WINTEST\_CHROME"*>

<parameter name=*"browser"* value=*"chrome"* />

<classes>

<class name=*"com.flightbooking.runner.TestRunner"* />

</classes>

</test> </suite>

**Pom.xml**

**Path: /FlightBooking2/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>com.automation.flightbooking</groupId>

<artifactId>FlightBooking</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>info.cukes</groupId>

<artifactId>cucumber-java</artifactId>

<version>1.2.5</version>

</dependency>

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>6.14.3</version>

</dependency>

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

<dependency>

<groupId>info.cukes</groupId>

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

<version>1.0.5</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>info.cukes</groupId>

<artifactId>cucumber-testng</artifactId>

<version>1.2.5</version>

<exclusions>

<exclusion>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

</exclusion>

</exclusions>

</dependency>

<dependency>

<groupId>info.cukes</groupId>

<artifactId>cucumber-core</artifactId>

<version>1.2.5</version>

</dependency>

<dependency>

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

<artifactId>selenium-java</artifactId>

<version>2.53.0</version>

</dependency>

<!-- https://mvnrepository.com/artifact/log4j/log4j -->

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.17</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

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

<version>3.8.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

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

<version>3.0.0-M5</version>

<configuration>

<suiteXmlFiles>

<suiteXmlFile>src\test\resources\TestNG\testng.xml</suiteXmlFile>

</suiteXmlFiles>

</configuration>

</plugin>

</plugins>

</build>

</project>

**log4j.xml**

**Path: /FlightBooking2/src/test/java/com/flightbooking/utility/log4j.xml**

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

<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">

<log4j:configuration xmlns:log4j=*"https://jakarta.apache.org/log4j/"* debug=*"false"*>

<appender name=*"fileAppender"* class=*"org.apache.log4j.FileAppender"*>

<param name=*"Threshold"* value=*"INFO"* />

<param name=*"File"* value=*"logfile.log"*/>

<layout class=*"org.apache.log4j.PatternLayout"*>

<param name=*"ConversionPattern"* value=*"%d %-5p [%c{1}] %m %n"* />

</layout>

</appender>

<root>

<level value=*"INFO"*/>

<appender-ref ref=*"fileAppender"*/>

</root>

</log4j:configuration>

**Log.java**

**Path: /FlightBooking2/src/test/java/com/flightbooking/utility/Log.java**

**package** com.flightbooking.utility;

**import** org.apache.log4j.Logger;

**public** **class** Log {

//Initialize Log4j logs

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

// This is to print log for the beginning of the test case, as we usually run so many test cases as a test suite

**public** **static** **void** startTestCase(String sTestCaseName){

*Log*.info("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

*Log*.info("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

*Log*.info("$$$$$$$$$$$$$$$$$$$$$ "+sTestCaseName+ " $$$$$$$$$$$$$$$$$$$$$$$$$");

*Log*.info("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

*Log*.info("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

}

//This is to print log for the ending of the test case

**public** **static** **void** endTestCase(String sTestCaseName){

*Log*.info("XXXXXXXXXXXXXXXXXXXXXXX "+"-E---N---D-"+" XXXXXXXXXXXXXXXXXXXXXX");

*Log*.info("X");

*Log*.info("X");

*Log*.info("X");

*Log*.info("X");

}

// Need to create these methods, so that they can be called

**public** **static** **void** info(String message) {

*Log*.info(message);

}

**public** **static** **void** warn(String message) {

*Log*.warn(message);

}

**public** **static** **void** error(String message) {

*Log*.error(message);

}

**public** **static** **void** fatal(String message) {

*Log*.fatal(message);

}

**public** **static** **void** debug(String message) {

*Log*.debug(message);

}

}