A Micro Project report on

# Perform Automation Testing For Any Hotel Booking Website

Submitted to the CMR Institute of Technology in partial

Submitted to the CMR Institute of Technology in partial fulfillment of the requirement for the award of the Laboratory of

**Automated Testing Tool Lab (20-CS-PC-317)**

**of**

**III-B.Tech. I-Semester**

**in**

**Computer Science and Engineering Department**

Submitted by


| | |
|---|---|
| **B.GANESH SAI** | **(20R01A0507)** |
| **T. VARUN KUMAR** | **(20R01A0555)** |
| **T. SRINIDHI** | **(20R01A0556)** |
| **V. RAJU** | **(20R01A0557)** |
| **B. VASANTH** | **(20R01A0558)** |
| **V. BHAVISHYA** | **(20R01A0559)** |
| **Y. BHASKAR** | **(20R01A0560)** |

Under the Guidance Of

**Mr. N. ABHISHEK**

Assistant Professor, Department of Computer Science and Engineering




**CMR INSTITUTE OF TECHNOLOGY**

**(UGC AUTONOMOUS)**

**(Approved by AICTE,Affiliated to JNTU, Kukatpally, Hyderabad)**

**Kandlakoya, Medchal Road, Hyderabad**


**2022-2023**

# CMR INSTITUTE OF TECHNOLOGY

**(UGC AUTONOMOUS)**

**(Approved by AICTE, Affiliated to JNTU, Kukatpally, Hyderabad)**

**Kandlakoya, Medchal Road, Hyderabad.**

## Department of Computer Science and Engineering


**CERTIFICATE**

This is to certify that a Micro Project entitled with: **"Perform Automation Testing For Any Hotel Booking Website"** is Being

Submitted By

| | |
|---|---|
| **B. GANESH SAI** | **(20R01A0507)** |
| **T. VARUN KUMAR** | **(20R01A0555)** |
| **T. SRINIDHI** | **(20R01A0556)** |
| **V. RAJU** | **(20R01A0557)** |
| **B. VASANTH** | **(20R01A0558)** |
| **V. BHAVISHYA** | **(20R01A0559)** |
| **Y. BHASKAR** | **(20R01A0560)** |

In partial fulfillment of the requirement for award of the Automated Testing Tool Lab (20-CS-PC-317) of III-B.Tech I- Semester in Department of Computer Science and Engineering towards a record of a bonafide work carried out under our guidance and supervision.

**Signature of Faculty**
**(Mr.N.Abhishek)**

**Signature of HOD**
**(Mr. A. Prakash)**

**Course Coordinator**

# ACKNOWLEDGEMENT

We are extremely grateful to **Dr. M. Janga Reddy**, **Director**, **Dr. B. Satyanarayana**,**Principal** and **Mr. A. Prakash**, **Head of Department**, Department of Computer Science and Engineering, CMR Institute of Technology for their inspiration and valuable guidance during entire duration.

We are extremely thankful to our Automated Testing Tool Lab faculty in-charge, **Mr.N.ABHISHEK,** Assistant Professor, Department of Computer Science and Engineering, CMR Institute of Technology for his constant guidance, encouragement and moral support throughout the project.

We express our thanks to all staff members and friends for all the help and coordination extended in bringing out this Project successfully in time.

Finally, we are very much thankful to our parents and relatives who guided directly or indirectly for successful completion of the project.

**B. GANESH SAI** **(20R01A0507)**
**T. VARUN KUMAR** **(20R01A0555)**
**T. SRINIDHI** **(20R01A0556)**
**V. RAJU** **(20R01A0557)**
**B. VASANTH** **(20R01A0558)**
**V. BHAVISHYA** **(20R01A0559)**
**Y. BHASKAR** **(20R01A0560)**

# CONTENTS

# AUTOMATED TESTING TOOLS SELENIUM

## INTRODUCTION TO SELENIUM :

Selenium is a set of open-source web automation tools that leverages the power of web browsers and helps in automating workflows of how users interact with the web application within the browser. The primary purpose of Selenium, as highlighted by the ***Selenium Website,*** is - "**Selenium Automates Browsers, What you do with this power is entirely up to you**". Even with the emerging tools and technologies, Selenium is still leading the board in the list of Web Automation tools and Automation Testing.

## SELENIUM TESTING TOOLS,COMPONENTS AND THEIR USES :

Selenium has been in the industry for a long time and used by automation testers all around the globe.
Let's check the four major components of Selenium –

- Selenium IDE
- Selenium RC
- Selenium Web driver
- Selenium GRID

### Selenium IDE

Selenium IDE (Integrated Development Environment) is the major tool in the Selenium Suite. It is a complete integrated development environment (IDE) for Selenium tests. It is implemented as a Firefox Add-On and as a Chrome Extension. It allows for recording, editing and debugging of functional tests. It was previously known as Selenium Recorder. Selenium-IDE was originally created by Shinya Kasatani and donated to the Selenium project in 2006. Selenium IDE was previously little-maintained. Selenium IDE began being actively maintained in 2018.

Scripts may be automatically recorded and edited manually providing autocompletion support and the ability to move commands around quickly. Scripts are recorded in Selenese, a special test scripting language for Selenium. Selenese provides commands for performing actions in a browser (click a link, select an option) and for retrieving data from the resulting pages.

### Selenium RC (Remote control)

Selenium Remote Control (RC) is a server, written in Java, that accepts commands for the browser via HTTP. RC makes it possible to write automated tests for a web application in any programming language, which allows for better integration of Selenium in existing unit test frameworks. To make writing tests easier, Selenium project currently provides client

drivers for PHP, Python, Ruby, .NET, Perl and Java. The Java driver can also be used with JavaScript (via the Rhino engine). An instance of selenium RC server is needed to launch html test case – which means that the port should be different for each parallel run. However, for Java/PHP test case only one Selenium RC instance needs to be running continuously.

## Selenium Web Driver

Selenium WebDriver is the successor to Selenium RC. Selenium WebDriver accepts commands (sent in Selenese, or via a Client API) and sends them to a browser. This is implemented through a browser-specific browser driver, which sends commands to a browser and retrieves results. Most browser drivers actually launch and access a browser application (such as Firefox, Google Chrome, Internet Explorer, Safari, or Microsoft Edge); there is also an HtmlUnit browser driver, which simulates a browser using the headless browser HtmlUnit.

Selenium WebDriver does not need a special server to execute tests. Instead, the WebDriver directly starts a browser instance and controls it. However, Selenium Grid can be used with WebDriver to execute tests on remote systems (see below). Where possible, WebDriver uses native operating system level functionality rather than browser-based JavaScript commands to drive the browser. This bypasses problems with subtle differences between native and JavaScript commands, including security restrictions.

## Selenium GRID

Selenium Grid is a server that allows tests to use web browser instances running on remote machines. With Selenium Grid, one server acts as the hub. Tests contact the hub to obtain access to browser instances. The hub has a list of servers that provide access to browser instances (WebDriver nodes), and lets tests use these instances. Selenium Grid allows running tests in parallel on multiple machines and to manage different browser versions and browser configurations centrally (instead of in each individual test).

The ability to run tests on remote browser instances is useful to spread the load of testing across several machines and to run tests in browsers running on different platforms or operating systems. The latter is particularly useful in cases where not all browsers to be used for testing can run on the same platform.

# HOW TO OPEN CHROME BROWSER USING SELENIUM :

Selenium is an open-source popular web-based automation tool. The major advantage of using selenium is, it supports all browsers like Google Chrome, Microsoft Edge, Mozilla Firefox, and Safari, works on all major OS and its scripts are written in various languages i.e Java, Python, JavaScript, C#, etc. We will be working with Java. In this article, let us consider a test case in which we will try to automate the following scenarios in the Google Chrome browser.

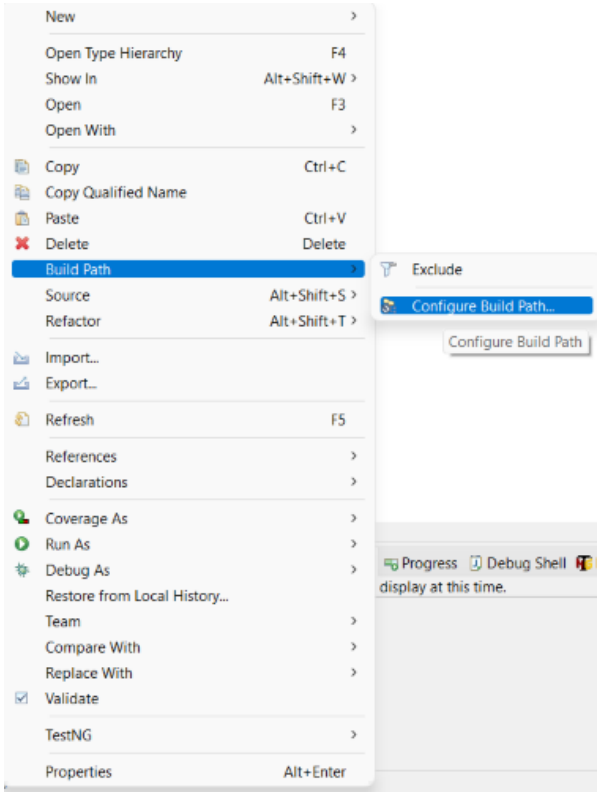
- Launch Chrome browser.
- Maximize the browser.
- Open URL: https://www.google.com

For invoking the chrome browser, we need the Eclipse IDE, Selenium Grid(version 4), and Chrome web Driver.

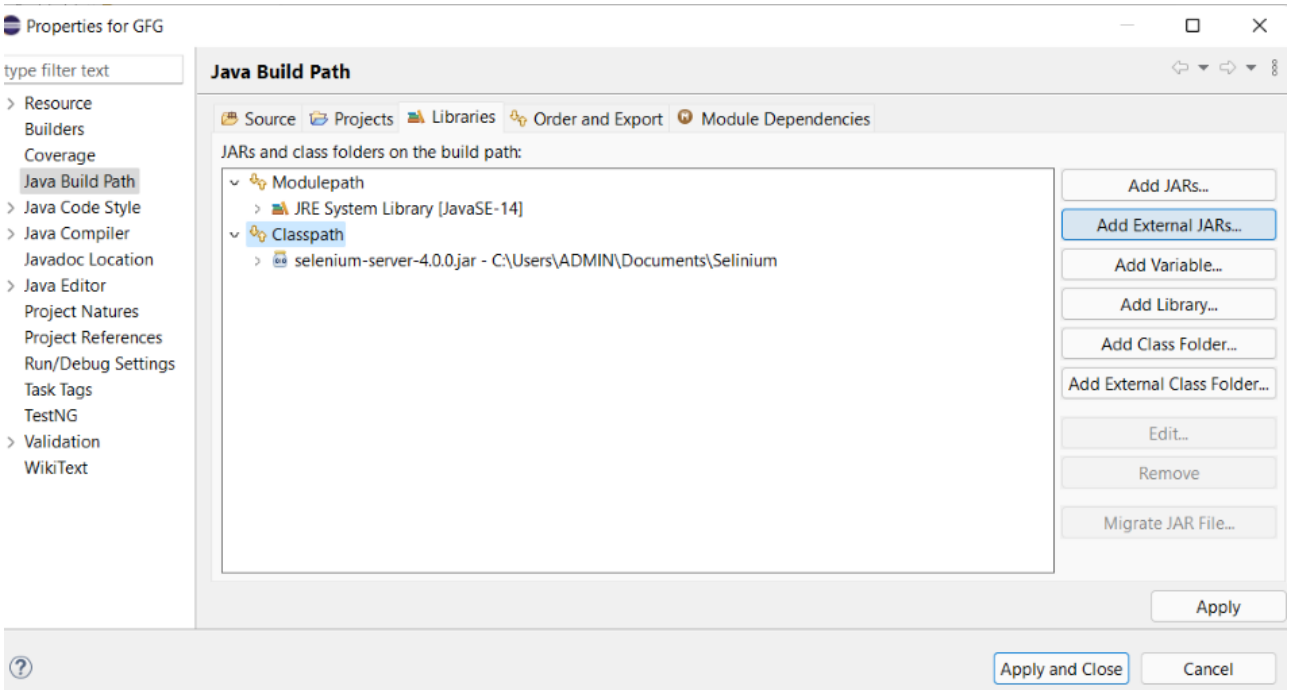
## Step by Step Implementation :

Step 1: Open the Eclipse IDE and create a new Java project. Right-click on the "src" folder and create a new Class File from New > Class. Give the Class name and click on the "Finish" button.

Step 2: Add Selenium JAR file into the Java Project. Right-click on Class name and Select "Build Path" and select > configure build path.



Then select Libraries > Classpath > and Click “Add External JAR’s”, now add the Selenium Jar and click “Apply and Finish”.



## USE OF SELENIUM CHROMEDRIVER:

The most common web browser used today is Google Chrome. Google Chrome is widely used globally, making it essential for all the websites on Chrome to be tested. This article on ChromeDriver in Selenium will help you understand the basics of ChromeDriver and explain how a ChromeDriver is used for automated testing.

A ChromeDriver is a separate executable or a standalone server that Selenium WebDriver uses to launch Google Chrome. Here, a WebDriver refers to a collection of APIs used to automate the testing of web applications.

Initializing the object of ChromeDriver is possible with the help of this command: ->WebDriver driver = new ChromeDriver

As Google Chrome dominates the browser market, the use of a ChromeDriver becomes a must. Selenium WebDriver uses the ChromeDriver to communicate test scripts with Google Chrome. It is used to navigate between web pages and provide input to the same.

### Setting Up a Chrome Driver:

Four prerequisites are to be kept in mind when thinking of downloading and installing the Selenium ChromeDriver. They are as follows:

1. Java / JDK / JRE (https://www.oracle.com/in/java/technologies/javase-downloads.html)
2. Eclipse IDE (https://www.eclipse.org/downloads/)
3. Selenium (https://www.selenium.dev/)
4. Chrome Browser (https://www.selenium.dev/)

To learn how we can download and install Selenium on our system.

## USE OF SELENIUM WEBDRIVER:

- It is one of the most popular Open-Source tools and is easy to get started with for testing web-based applications. It also allows you to perform cross browser compatibility testing.
- Supports multiple operating systems like Windows, Mac, Linux, Unix, etc.
- It provides compatibility with a range of languages including Python, Java, Perl, Ruby, etc.
- Provides support for modern browsers like Chrome, Firefox, Opera, Safari, and Internet Explorer.
- Selenium WebDriver completes the execution of test scripts faster when compared to other tools.
- More Concise API (Application Programming interface) than Selenium RC's.
- It also provides compatibility with iPhoneDriver, HtmlUnitDriver, and AndroidDriver.

### Limitations of WebDriver:

- Support for new browsers is not readily available when compared to Selenium RC
- For the automatic generation of test results, it doesn't have a built-in command

### How Selenium WebDriver Works:

On a high-level, Selenium WebDriver works in three steps:

- Test commands are converted into an HTTP request by the JSON wire protocol.
- Before executing any test cases, every browser has its own driver which initializes the server.
- The browser then starts receiving the request through its driver.

Let’s take an example with the code snippet below:

```
WebDriver driver = new ChromeDriver ();

driver. get (https://www.browserstack.com)
```

As soon as you complete writing your code, execute the program. The above code will result in the launching of the Chrome browser which will navigate to the BrowserStack website.
Now let us understand what goes behind the scene when you click on Run until the launching of the Chrome Browser.
Once the program is executed, every line of code/script will get transformed into a URL. The JSON Wire protocol over HTTP makes this possible. Then this URL is passed to the browser drivers (in our example, the ChromeDriver). At this point, our client library (Python in our example) translates the code into JSON format and interacts with the ChromeDriver.

**The URL after JSON conversion looks as follows:**

```
https://localhost:8080/{"url":https://www.browserstack.com"}
```

To receive the HTTP requests, every Browser Driver uses an HTTP server. Once the browser driver receives the URL, it processes the request by passing it to the real browser over HTTP. And then all your commands in the Selenium scripts will be executed.

**Use of HashMap in Selenium Automation Testing:**

Map is an interface in Java that works on a **Key and Value pair** concept, which cannot contain duplicate keys and each Key maps to, at the most, one value. HashMap is the implementation class of Map, which we will be using to store the data from the Excel sheet, which will be related to test steps of a particular testcase. First of all, we will just require a Java HashMap to extract the test steps. One more testcase has been added to show what happens when only a HashMap is used to extract multiple test case steps.

The following code shows how to extract data in HashMap from the Framework.xls:

```
 public static void main(String[] args) {    Fillo fillo = new Fillo();
Map<String, String> testcaseData = new HashMap<String, ...
```

**For the above we will write the following code :**

The above script is for opening Google Chrome Browser and navigating to geeksforgeeks website. So let's see how it works:

1. Set a system property "webdriver.chrome.driver" to the path of your ChromeDriver.exe file and instantiate a ChromeDriver class: *System.setProperty("webdriver.chrome.driver","chromedriver location");*
2. Maximize the window: *driver.manage().window().maximize();*
3. To open the URL: *driver.get("URL link")*

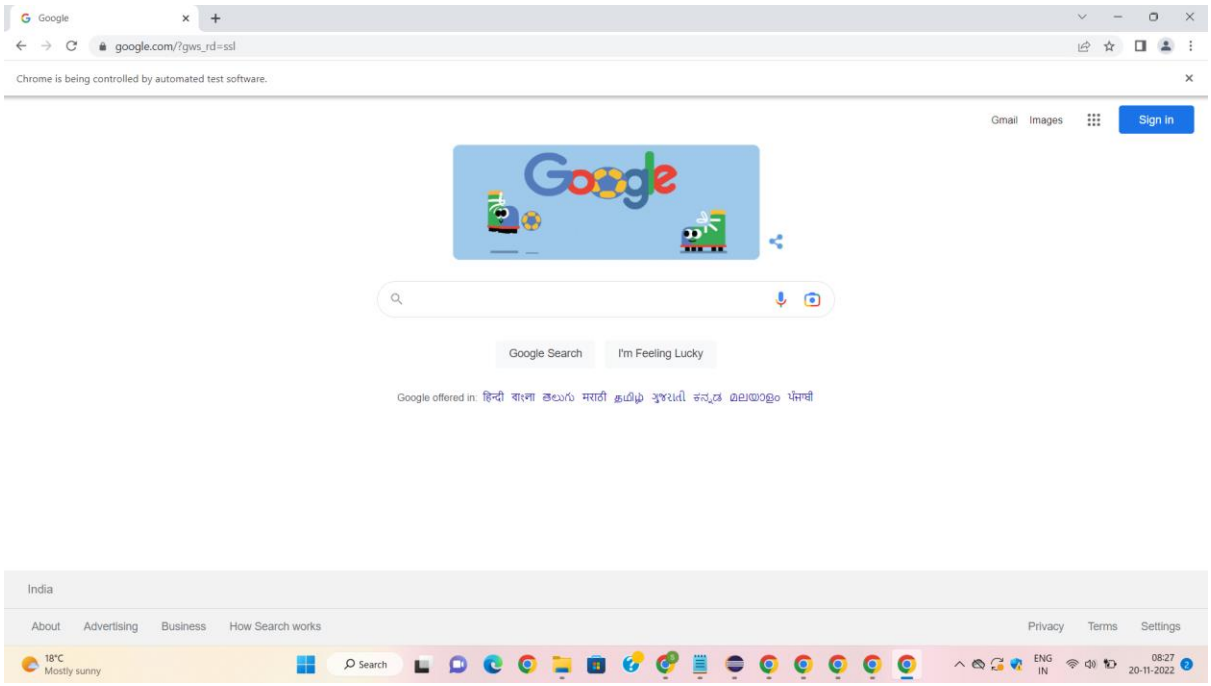
## SOURCE CODE:

```
package micro1;
import java.awt.AWTException;
import java.awt.Robot;
import java.awt.event.KeyEvent;

import org.openqa.selenium.By;
import org.openqa.selenium.Keys;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class Micro1 {
public static void main(String[] args) throws InterruptedException, AWTException {
System.setProperty("webdriver.chrome.driver","C:\\Users\\SWEETY\\Downloads\\chromedriver_win32(1) \\chromedriver.exe");

WebDriver driver = new ChromeDriver();
driver.manage().window().maximize();
driver.get("http://www.google.com");
WebElement gsearch = driver.findElement(By.name("q"));
gsearch.sendKeys("taj hotels");
gsearch.sendKeys(Keys.ENTER);

WebElement hotellink = driver.findElement(By.xpath("//span[text()='Taj Hotels - Book on Official Website']"));
hotellink.click();
WebElement book = driver.findElement(By.xpath("(//button[@class='cm-btn-secondary book-stay-btn'])[2]"));
book.click();
WebElement hotels = driver.findElement(By.xpath("(//input[@class='searchbar-input'])[1]"));
hotels.click();
hotels.sendKeys("hotels in hyderabad");
Thread.sleep(2000);
```

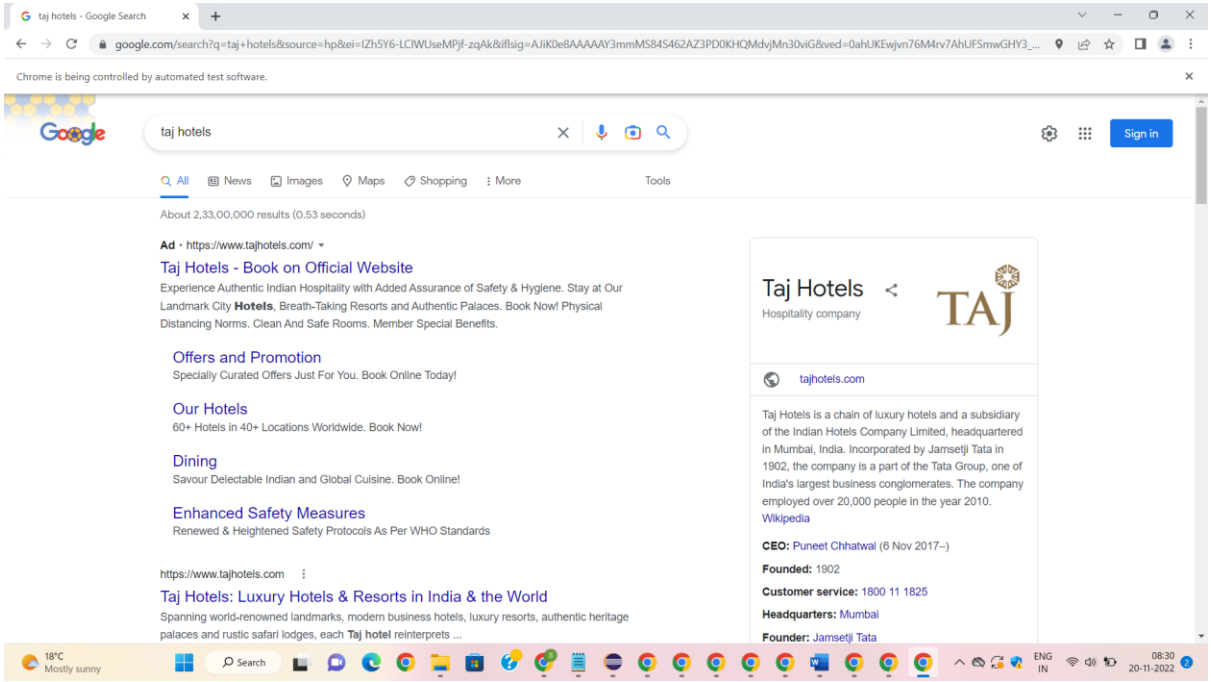
```
WebElement hotel = driver.findElement(By.xpath("//a[text()='Hotels In Hyderabad']"));
hotel.click();
WebElement add = driver.findElement(By.xpath("//button[@class='quantity-right-plus quantity-right-plus1 btn btn-default btn-number']"));
add.click();

WebElement chckavl = driver.findElement(By.xpath("//div[@class='mr-right best-avail-rate cm-btn-secondary best-avail-rate-enable']"));
chckavl.click();

WebElement viewHotel = driver.findElement(By.xpath("(//button[@class='cm-btn-secondary mr-list-view-hotels-button'])[1]"));
viewHotel.click();

WebElement viewDetails = driver.findElement(By.xpath("(//button[@class='cm-btn-secondary'])[7]"));
viewDetails.click();
Thread.sleep(5000);
WebElement viewRates = driver.findElement(By.xpath("(//button[@class='more-rates-button cm-btn-secondary'])[1]"));
viewRates.click();

WebElement selectRoom = driver.findElement(By.xpath("(//button[@class='cm-btn-secondary more-rates-select-btn'])[1]"));
selectRoom.click();

//details
Thread.sleep(3000);

WebElement drp = driver.findElement(By.xpath("//span[@class='icon-drop-down-arrow']"));
drp.click();
Robot r = new Robot();
r.keyPress(KeyEvent.VK_DOWN);
Thread.sleep(1000);
r.keyPress(KeyEvent.VK_ENTER);
WebElement name = driver.findElement(By.name("guestFirstName"));
name.sendKeys("John");
```

```
WebElement lname = driver.findElement(By.name("guestLastName"));
lname.sendKeys("Wick");

WebElement email = driver.findElement(By.name("guestEmail"));
email.sendKeys("johnwick123@gmail.com");

WebElement ph = driver.findElement(By.name("guestPhoneNumber"));
ph.sendKeys("9876543210");

WebElement terms1 = driver.findElement(By.xpath("(//span[@class='mr-inputCustomShape'])[3]"));
terms1.click();

WebElement terms2 = driver.findElement(By.xpath("(//span[@class='mr-inputCustomShape'])[4]"));
terms2.click();

WebElement conf = driver.findElement(By.xpath("//button[@class='pay-now-button cm-btn-secondary']"));
conf.click();
}
}
```

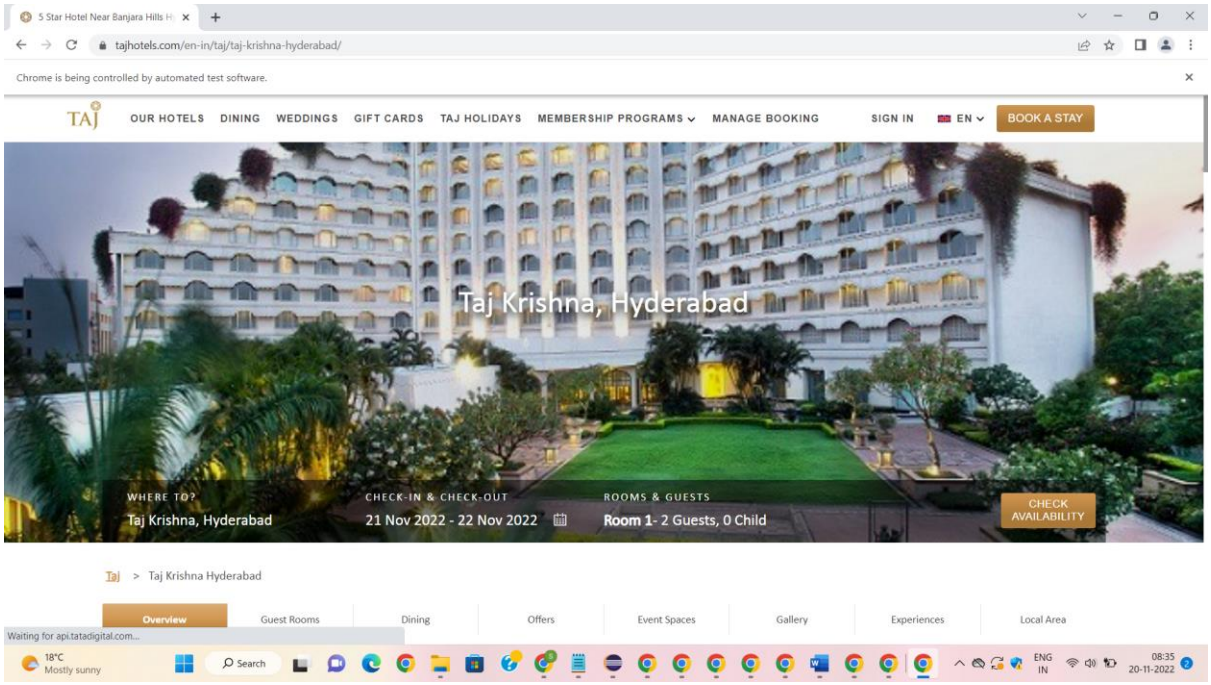
# OUTPUT:



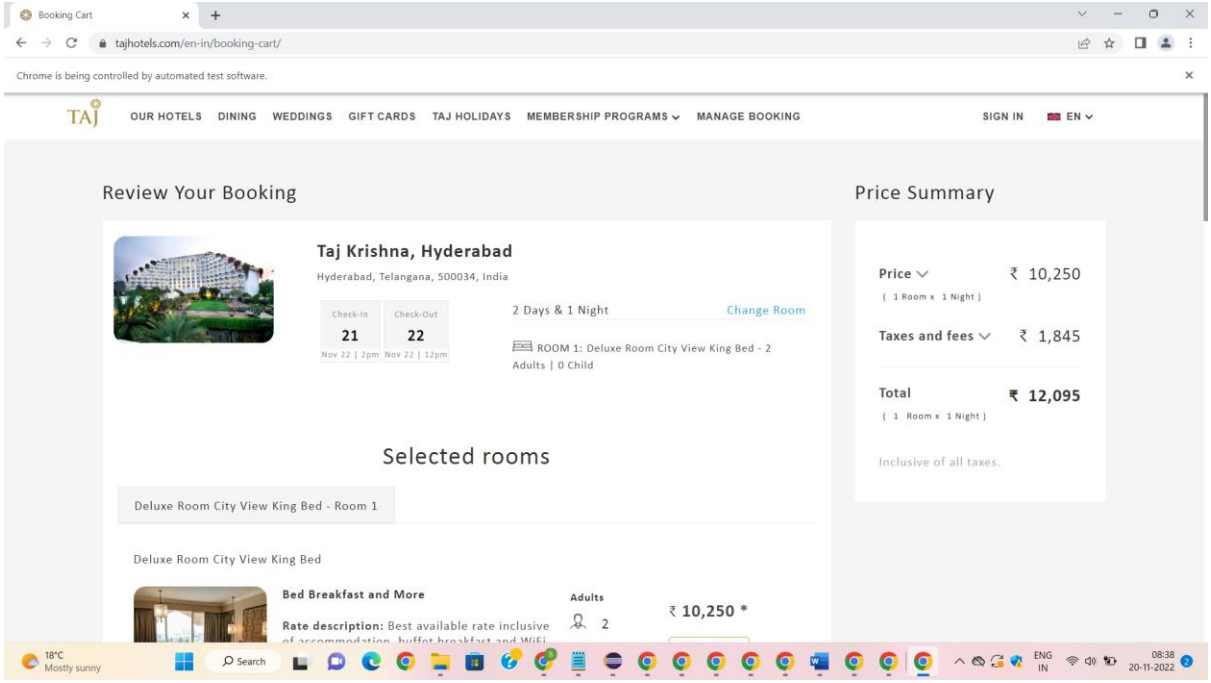

First we search for google



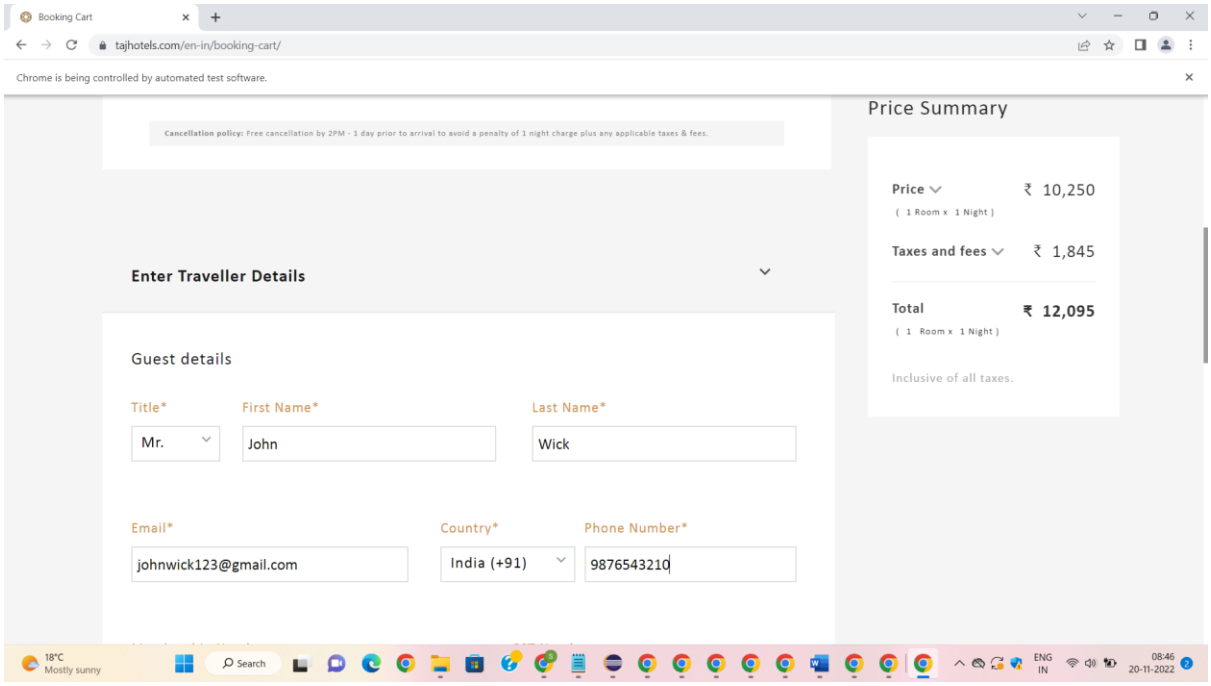
After it  was clicked then we get many links related to it .We selected the first link from all those.



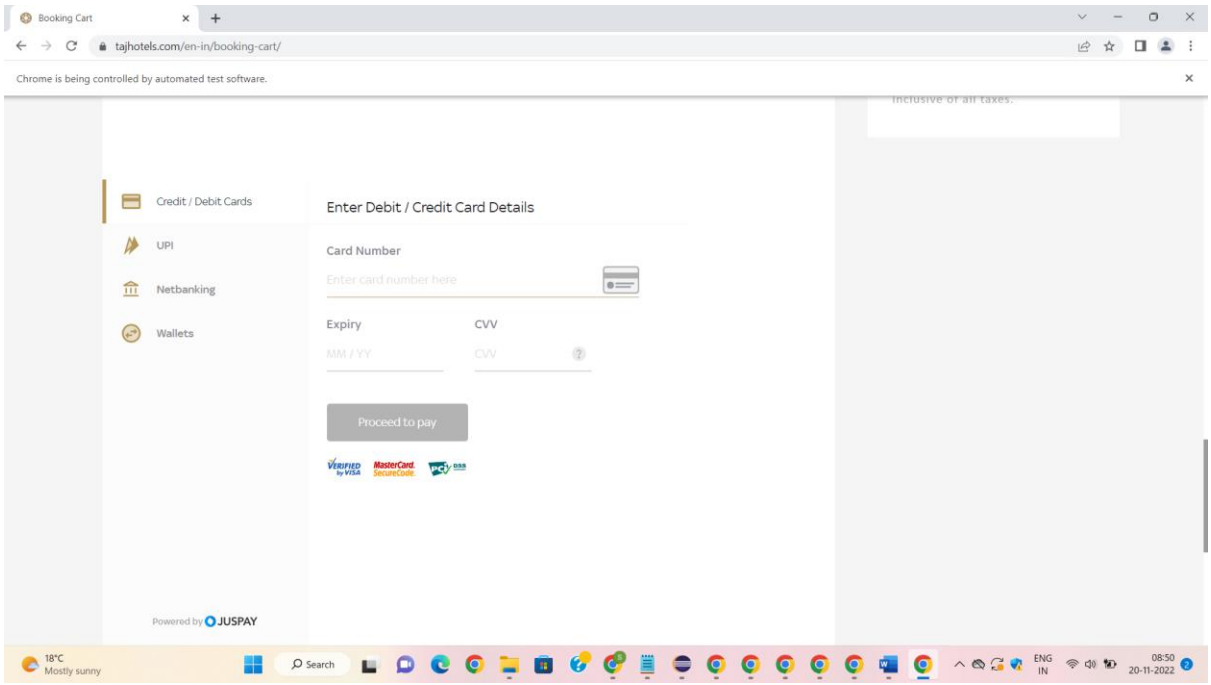

We choose the Taj Krishna,Hyderabad




Then we booked Deluxe Room City View King Bed




We enter our details for Booking Confirmation




Finally we paid money through Debit card

## CONCLUSION:

Our report is on Automation testing for finding the availability of hotel booking from taj hotel website. First we enter to google searchbar then we enter taj hotel then login to the website. After it was clicked then we get many links related to it. We select the taj hotel link. As we follow the above steps we can find the availability of rooms and then book.

## REFERENCES:


1. https://www.functionize.com/selenium-testing
2. http://www.helloselenium.com/2017/02/how-to-set-java-path-for-windows-10.html
3. https://chromedriver.chromium.org/downloads
4. https://www.tajhotels.com/?gclid=Cj0KCQiAj4ecBhD3ARIsAM4Q_jG9UBvwnz8U2j3uhkIefMkUupN6MyuYdjr6g-muNJmu5Je5o7ye21UaAs_3EALw_wcB