SELENIUM

**Research on Automation Testing Tool - Selenium**

| **Submitted By:** | **Submitted To:** |
| --- | --- |
| **Nischal Shrestha** | **Prajesh Shakya** |

**Date: 20th May, 2025**

# ABSTRACT

This document provides a comprehensive overview of Selenium, a powerful open-source suite of tools and libraries for automating web browser interactions. It details Selenium’s capabilities in mimicking user behavior, supporting the W3C WebDriver standard, and facilitating cross-browser, cross-language, and cross-platform testing. This document delves into its four component architecture (Client Library, JSON Wire Protocol, Browser Drivers, Browsers) and highlights its three primary components: Selenium WebDriver, Selenium IDE, and Selenium Grid. It also explores popular frameworks, user demographics, and the significant benefits of using Selenium for automation testing, such as its open-source nature, multilingual support, and seamless integration into CI/CD pipelines.

# TABLE OF CONTENTS

[**ABSTRACT 2**](#_g89e2vk28p4q)

[**TABLE OF CONTENTS 3**](#_g3io5mhx9ze4)

[**SELENIUM 4**](#_3th4jkigretw)

[Introduction 5](#_gppux02x182z)

[Selenium Architecture 6](#_oqpuukkk2qit)

[Key Components of the Selenium Suite 7](#_9qvqil78jn51)

[1. Selenium WebDriver (The Heart of Selenium): 7](#_i90nm4gnrr3m)

[2. Selenium IDE (Integrated Development Environment): 7](#_wus85twtdpho)

[3. Selenium Grid: 8](#_wv8c36pqo8rp)

[Popular Frameworks for Selenium Testing 8](#_q98y95yry7ha)

[Who uses Selenium? 9](#_wvhtm67ioafq)

[1. Software Testers and QA Professionals 9](#_i9g4s4i98lc0)

[2. Developers 9](#_ze2qqqvk9qyi)

[3. DevOps Engineers 9](#_hgg22k5ixuhs)

[4. Automation Engineers 10](#_kfthqdgb7gwf)

[5. Freelancers and Consultants 10](#_qrfsxng9nu92)

[6. Educational Institutions and Students 10](#_lemr7igdnhov)

[Why use Selenium for Automation Testing? 10](#_o08szhjil27g)

[1. Open-Source 11](#_xnjt8ib71kav)

[2. Multilingual Support 11](#_orw4ugmuo8n)

[3. Cross-Platform Compatibility 11](#_j5q58lkjkwu)

[4. Cross Browser Testing 11](#_mvq3v53y1jg5)

[5. Framework Integration 11](#_94ugic5dz5k4)

[6. Parallel Testing 11](#_jwr76qr0zpp6)

[7. Mouse and Keyboard Simulation 12](#_tak3ode31r5u)

[8. Headless Browser Testing 12](#_gu92vc2u6luf)

[9. Continuous Integration and Continuous Deployment (CI/CD) 12](#_sn8ehui0dd0t)

[10. Comprehensive Reporting and Documentation 12](#_h47d7sst44uy)

[11. Community Support 12](#_e1rf21k788zj)

[Types of Testing Selenium can automate 13](#_vc2acjufy0q0)

[1. Functional Testing 13](#_e9ctzh1tqd2)

[2. Regression Testing 13](#_irbrn3cht0db)

[3. Visual Testing 13](#_sr89mm4uqw6x)

[4. Smoke Testing 13](#_7sp5qg9zvwzv)

[5. Cross Browser Testing 14](#_hg48zttspe8t)

[6. Data-Driven Testing 14](#_4y8njp7r4h43)

[7. UI Testing 14](#_6o57mnhlb78g)

[8. Monkey Testing 14](#_szed2qgndxxa)

[How Selenium Works (in simplified terms): 14](#_9r0x5u3tzmkr)

[Benefits of using Selenium: 15](#_y2itlbjrvzsr)

[**SELENIUM WEBDRIVER-INSTALLATION 16**](#_ijm5rruu5odt)

[Installation Steps 16](#_qgpa6f6rbly3)

[Implementation 16](#_bo36w5sj6wt7)

[Step 1: Setting up Environment 16](#_pvh4mfyrtzqm)

[Step 2: Downloading Selenium WebDriver Libraries 21](#_hi6b368aorie)

[Step 3: Download Browser Driver (e.g., Chrome) 22](#_1triq1jnlrom)

[Step 4: Downloading the Selenium WebDriver jar 24](#_xcstjsax0erg)

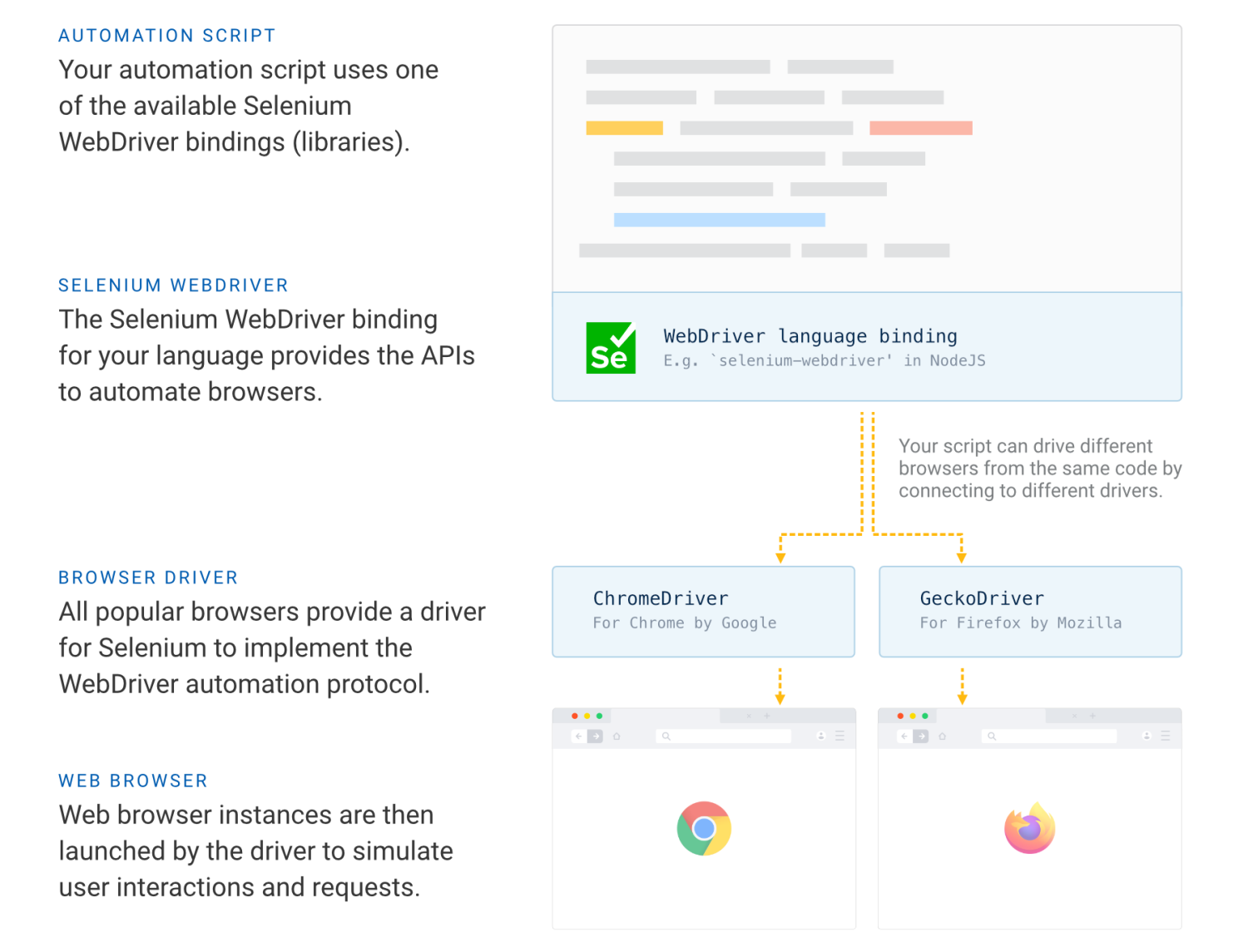
[Step 5: Creating Java Project 25](#_mco0ioarbgqs)

[**CONCLUSION 30**](#_myusf1rphev6)

# SELENIUM

## Introduction

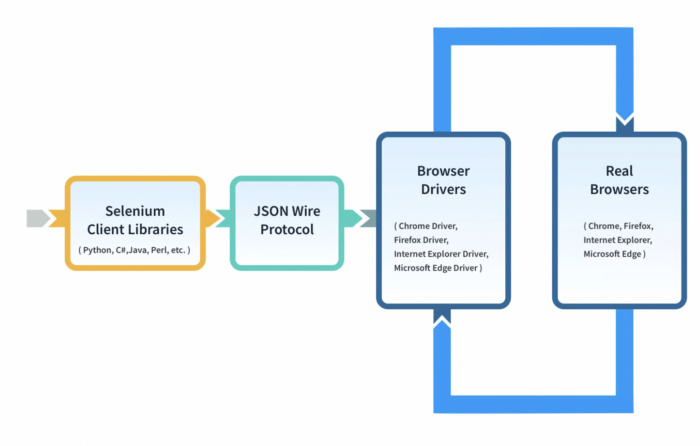
Selenium is a comprehensive suite offering various tools and libraries to automate web browser interactions. It includes features that mimic how users interact with browsers, and a server to manage browser usage across different platforms for testing purposes, but also for other tasks like data scraping, repetitive form filling, or managing online accounts. It supports the W3C WebDriver standard that helps testers write code to make it work across all the major web browsers. With Selenium, developers and testers can automate website testing, including functional testing, regression testing, and end-to-end testing. Its versatility in testing across different environments is attributed to its cross-browser, cross-language, and cross-platform capabilities. It seamlessly integrates with existing development workflows and supports a wide range of programming languages, including Java, JavaScript, C#, PHP, Python, and Ruby. Selenium offers extensive browser compatibility with major web browsers like Chrome, Firefox, Safari, Edge, and Opera to ensure comprehensive browser coverage. Its flexibility is further enhanced by its compatibility with different automation testing frameworks like TestNG, JUnit, MSTest, pytest, WebdriverIO, and more.



## Selenium Architecture

Selenium WebDriver Architecture is made up of four major components:

1. Selenium Client library: Selenium provides support to multiple libraries such as Ruby, Python, Java, etc as language bindings
2. JSON wire protocol over HTTP: JSON is an acronym for JavaScript Object Notation. It is an open standard that provides a transport mechanism for transferring data between client and server on the web.
3. Browser Drivers: Selenium browser drivers are native to each browser, interacting with the browser by establishing a secure connection. Selenium supports different browser drivers such as ChromeDriver, GeckoDriver, Microsoft Edge WebDriver, SafariDriver, and InternetExplorerDriver.
4. Browsers: Selenium provides support for multiple browsers like Chrome, Firefox, Safari, Internet Explorer etc.



## Key Components of the Selenium Suite

Selenium comprises three primary components:

### Selenium WebDriver (The Heart of Selenium):

It is a component of the Selenium suite for automating web application testing. It provides an interface for developers and testers to create and run test scripts, which simulate user interactions with web browsers. This includes actions like clicking on links, filling out forms, and fetching web page data. All of these actions can be done either on local machines or remotely.

When running Selenium tests on a remote (cloud) grid, there comes Selenium RemoteWebDriver that allows for running browser automation tests on a machine hosted on the cloud. Basically, it’s a version of the WebDriver but runs on a remote server. Specific browser drivers, such as FirefoxDriver, ChromeDriver, and InternetExplorerDriver, are all built on top of the RemoteWebDriver class. This means they share its capabilities but are tailored to their respective browsers.

With Selenium’s version 4.6.0, Selenium Manager was launched, which handles the browser drivers under the hood, so you don’t have to worry about the browser drivers. It streamlines the setup process for Selenium users. This means that you no longer need to manually download, set up, or update browser drivers for different browsers like Chrome, Firefox, or Edge. Instead, Selenium Manager takes care of these aspects, ensuring that the appropriate browser drivers are always in use and up to date.

It supports all major browsers like Chrome, Firefox, Safari, Edge, and Internet Explorer. Selenium WebDriver is the language bindings and implementations of specific browser-controlling codes. In June 2018, WebDriver received a significant endorsement as it became a W3C WebDriver Protocol.

### Selenium IDE (Integrated Development Environment):

Selenium IDE is an extension for Chrome, Firefox, and Edge that streamlines the task of recording and executing tests directly within the browser. With its intuitive interface and pre-built functionality, it enables testers to create reliable tests without any additional setup requirements quickly.

Moreover, it offers advanced debugging capabilities such as breakpoints and exception pausing, making the troubleshooting process more efficient. Another advantage of using Selenium IDE is its support for automated cross browser testing. This means you can run your tests on different browsers and operating systems using the Command-line Runner.

### Selenium Grid:

Selenium Grid is a part of the Selenium suite for the running of multiple test scripts across different browsers, operating systems, and machines simultaneously. It uses a hub-and-node architecture, where the hub acts as a central point to control the network of test machines (nodes).

This setup enables distributed testing, significantly reducing the time for cross-browser and cross-platform testing. Grid facilitates parallel execution of tests, enhancing testing efficiency and coverage. It is particularly useful for large-scale test environments and continuous integration pipelines.

The stable release of Selenium 4, released in October 2021, brings major updates from its predecessor, Selenium 3. A key feature in Selenium 4 is the adoption of the W3C WebDriver Protocol, which replaces the JSON Wire Protocol used in Selenium 3. In this new version, the tester does not need to initiate Hub and Node separately to perform automation testing. Instead, Selenium 4 integrates the Hub and Node into a single jar file.

The architecture of Selenium Grid 4 supports four main processes: Session Map, Node, Router, and Distributor. Additionally, the Selenium 4 WebDriver hierarchy offers several enhanced features, including an improved graphical user interface and built-in Docker support, further augmenting its capabilities for automation testing.

## Popular Frameworks for Selenium Testing

| Selenium Supported Language | Framework/Library |
| --- | --- |
| Selenium Java | Selenide, Gauge, TestNG, Geb, JUnit, Cucumber, Serenity BDD |
| Selenium Python | Unittest, pytest, Behave, Robot, Lettuce |
| Selenium JavaScript | WebdriverIO, [Nightwatch.js](http://nightwatch.js), Mocha, Jest, AngularJS, Cucumber, Jasmine, Karma, TestCafe, [Nemo.js](http://nemo.js), Protractor |
| Selenium C# | SpecFlow, MSTest, NUnit |
| Selenium Ruby | RSpec, Cucumber, test-unit, Capybara |
| Selenium PHP | Laravel, Codeception, PHPUnit, Behat |

## Who uses Selenium?

Selenium, a powerful open-source tool, has gained immense popularity in the field of automated testing due to its flexibility and versatility. The primary users of this automation testing framework are:

### Software Testers and QA Professionals

They leverage Selenium’s capabilities to automate web browsers and validate the functionality of web applications across different browsers and platforms. With Selenium, testers can execute repetitive test cases, perform regression testing, and ensure the reliability and performance of web applications.

### Developers

They also use Selenium to implement and validate the functionality of web features during the development phase. By automating browser actions and interactions, developers can identify and fix issues early in the development cycle, leading to faster delivery of high-quality software.

### DevOps Engineers

In the DevOps culture, continuous integration and continuous delivery (CI/CD) are crucial. DevOps engineers use Selenium to automate the testing process as part of the CI/CD pipeline. Selenium can be integrated with various DevOps tools to automate the deployment of web applications and ensure that new changes do not introduce regressions or break existing functionalities.

### Automation Engineers

They specialize in creating robust and maintainable test automation frameworks. They utilize Selenium to design and implement scalable automation solutions that can handle complex test scenarios and provide accurate test results. Selenium’s extensive support for multiple programming languages and its compatibility with various browsers and operating systems make it a preferred choice for automation engineers.

### Freelancers and Consultants

Freelancers and independent consultants who offer software testing and automation services also utilize Selenium. Its open-source nature and wide community support make it accessible and cost-effective, allowing freelancers to deliver high-quality testing solutions to clients without significant investments in proprietary tools.

### Educational Institutions and Students

They use Selenium for teaching and learning purposes. Many universities and training institutes include Selenium in their curriculum to equip students with practical skills in automated testing. Additionally, aspiring QA professionals and developers often use Selenium to enhance their technical proficiency and gain hands-on experience in automation testing.

## Why use Selenium for Automation Testing?

Automation Testing with Selenium involves using the Selenium suite of tools to automate web browsers for testing web applications. It enables developers and testers to write test scripts in various programming languages, run tests across multiple browsers and platforms, and ensure accuracy and efficiency in their testing processes. Some major reasons for choosing Selenium:

### Open-Source

Selenium is a free and open-source suite of tools, making it a cost-effective solution for web automation testing.

### Multilingual Support

It supports a wide range of programming languages, including Java, Python, PHP, C#, Ruby, and JavaScript. This flexibility allows testers to work with their preferred programming languages.

### Cross-Platform Compatibility

Selenium can be used on multiple operating systems, including Windows, macOS, and Linux, enabling testers to run tests on different platforms while ensuring consistent results across environments.

### Cross Browser Testing

Selenium supports automated testing on a variety of web browsers, including Chrome, Firefox, Safari, and more. This comprehensive coverage helps ensure that web applications function seamlessly across different browsers.

### Framework Integration

It can be integrated with various testing frameworks offered by Selenium-supported languages, such as TestNG, JUnit, TestCafe, and others. This integration streamlines the automation testing process by leveraging different capabilities of a framework.

### Parallel Testing

Parallel Testing with Selenium involves executing multiple test suites or test cases simultaneously to reduce the overall testing time. We can perform parallel testing either locally or on a cloud-bases grid, effectively reducing our software release cycles.

With its grid, we can run Selenium test scripts both on the local and cloud grid. Cloud testing facilitates the communication between the client (test script) and the remote browser instances, ensuring seamless execution of test commands on the desired browsers.

### Mouse and Keyboard Simulation

With Selenium, we can simulate real user interactions by replicating mouse behaviors and keyboard events. This is particularly useful for testing complex user interactions and scenarios such as automated interaction with web pages having dynamically loaded content, Single Page Applications (SPAs), and drag-and-drop interactions.

### Headless Browser Testing

Selenium lets us achieve faster test execution, optimize resource utilization, and enhance testing scalability. Headless browser testing is particularly well-suited for end-to-end testing, regression testing, performance testing, visual regression testing, and data extraction (web scraping).

### Continuous Integration and Continuous Deployment (CI/CD)

Whether tests are run locally or on a cloud grid, with Selenium, testers can easily trigger continuous tests in a CI pipeline by leveraging popular CI/CD tools like Jenkins, CircleCI, and Travis CI.

### Comprehensive Reporting and Documentation

Selenium provides detailed test execution logs and reports, making it easier for teams to track test results and pinpoint areas that require attention. This robust documentation ensures transparency and accountability throughout the testing process, facilitating better communication and collaboration among team members.

### Community Support

Selenium has a large and active community that provides extensive support and resources. This community-driven approach ensures that Selenium remains up-to-date and adapts to evolving web technologies.

## Types of Testing Selenium can automate

### Functional Testing

In functional testing, Selenium is used to validate if the website functions according to the specified requirements. It automates the execution of tests that check specific requirements. It automates the execution of tests that check specific functionalities, simulating user interactions like clicking, typing, and navigation. Since Selenium can interact with web elements and verify their responses, it is an ideal framework for testing the functional aspects of a website, including forms, dialogs, and other interactive features.

### Regression Testing

Regression Testing with Selenium involves re-running existing tests to ensure that new changes have not impacted the existing functionalities. This type of testing is crucial for continuous development environments where frequent changes are made to the website or web application. Selenium automates these tests, providing quick feedback on the impact of recent code updates and helps maintain the stability of the website over time.

### Visual Testing

Visual Testing using Selenium automates the testing of verifying that a website or web application looks and behaves as intended and there are no unintended visual appearance or functionality changes. In this process, a baseline version of a website or web application is compared with a current version to identify visual deviations.

### Smoke Testing

Smoke Testing with Selenium runs basic tests that validate the functions of websites. This is a quick testing phase that checks the health of the website after a new build or update goes live.

### Cross Browser Testing

Selenium excels in cross browser testing by running tests across multiple browsers, ensuring the website's behavior and appearance are consistent. This type of testing is important for web applications that target a broad user base, as it verifies compatibility with different browsers and their versions.

### Data-Driven Testing

In data-driven testing, Selenium is integrated with data handling tools to test websites under various data conditions. It processes multiple sets of input data, validating the website’s behavior in each scenario. This approach not only enhances test coverage but also ensures the robustness of the website across different data-driven use cases.

### UI Testing

UI Testing validates the visual elements of a web application. Automated UI testing with Selenium tests the UI components like buttons, text fields, and menus, ensuring they are not only present and visible but also functional and responsive. This testing verifies the usability and aesthetic aspects of the website.

### Monkey Testing

It involves breaking the system by giving a random input to the system. The input could be anything from an end-user’s perspective. It could be a click, scroll, or a dummy text string. Using Selenium, you can generate test scripts to automate monkey testing.

## How Selenium Works (in simplified terms):

1. **You write a test script:** Using a programming language and the Selenium WebDriver API, you write code that specifies actions you want to perform on a web page (e.g., driver.get("https://example.com"); driver.findElement(By.id("username")).sendKeys("testuser"); driver.findElement(By.id("loginButton")).click();).
2. **WebDriver sends commands:** When you run the script, WebDriver converts these commands into a format that the browser driver understands (using something like the JSON Wire Protocol over HTTP).
3. **Browser driver interacts with the browser:** The specific browser driver receives these commands and executes them directly on the browser.
4. **Browser performs actions:** The browser acts upon these commands, just as if a human user were interacting with it.
5. **Results are sent back:** The browser driver sends back the results of the actions (e.g., whether an element was found, the text of an element) to WebDriver, which then passes them back to your test script for assertions and reporting.

## Benefits of using Selenium:

* **Open-Source and Free:** No licensing costs, making it accessible to individuals and organizations of all sizes.
* **Cross-Browser Compatibility:** Supports all major browsers (Chrome, Firefox, Edge, Safari, Opera).
* **Multi-Language Support:** Allows writing test scripts in various popular programming languages.
* **Multi-Platform Support:** Works on Windows, macOS, Linux/Unix.
* **Parallel Testing (with Grid):** Significantly speeds up test execution by running tests concurrently.
* **Integration with Other Tools:** Seamlessly integrates with continuous integration/continuous delivery (CI/CD) tools (like Jenkins, GitLab), build tools (Maven, Gradle), and testing frameworks (TestNG, JUnit).
* **Flexibility and Reusability:** Test scripts can be highly customizable, reusable, and adapted to different testing scenarios.
* **Simulates Real User Actions:** Provides a high degree of confidence that the web application will function correctly for actual users.

# 

# SELENIUM WEBDRIVER-INSTALLATION

## Installation Steps

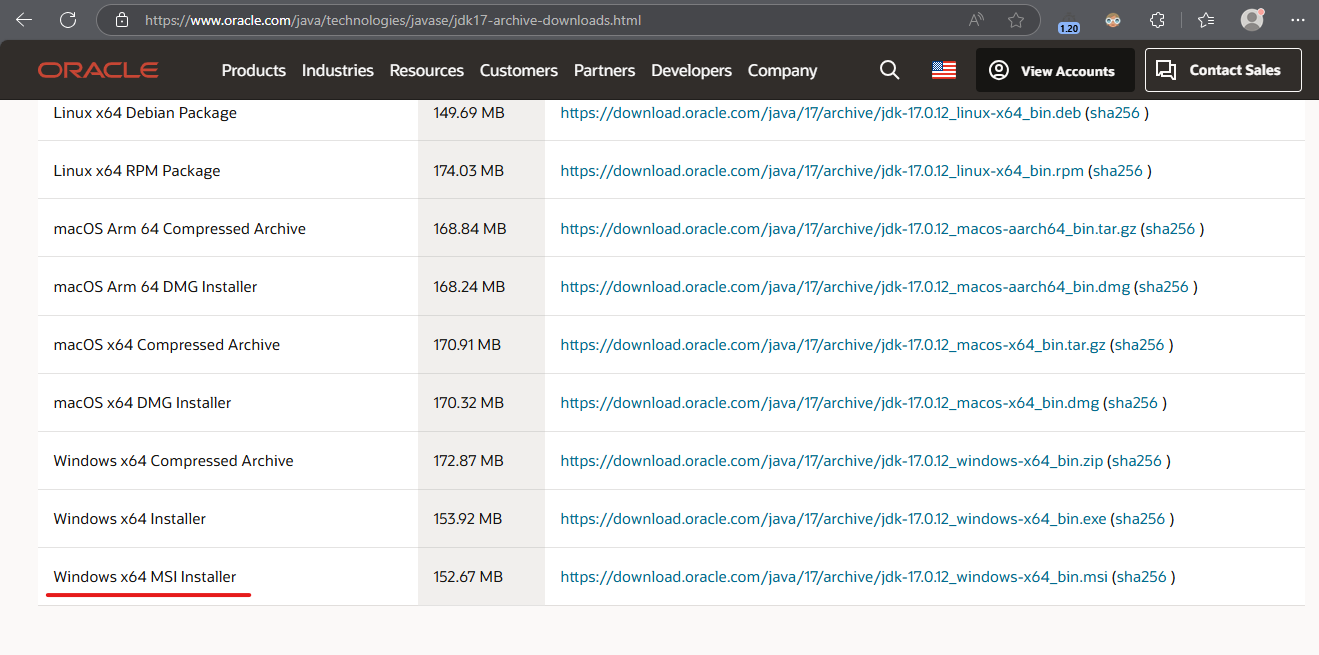
We have to follow the below steps to install the web driver in our system:

1. Download and Install Java
2. Download and setup Eclipse
3. Install the Selenium WebDriver Java Client
4. Configure Selenium WebDriver by adding the downloaded JAR files to our Java project in the IDE

## Implementation

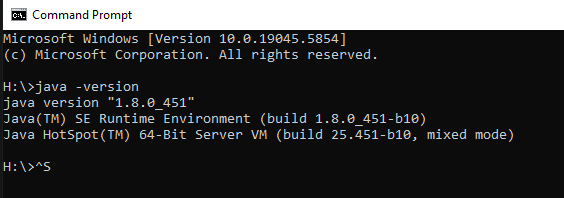
### Step 1: Setting up Environment

1. I installed Java 17 JDK on my system from “<https://www.oracle.com/java/technologies/javase/jdk17-archive-downloads.html>”.

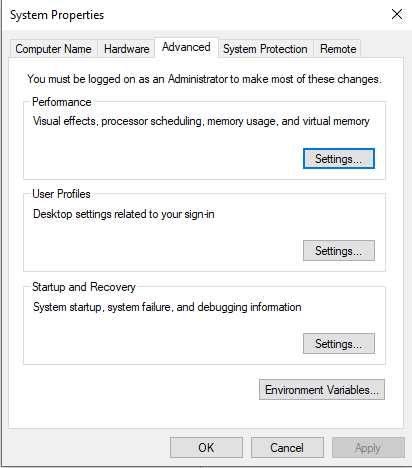


1. After installing Java on my system, I verified the installation by opening a command prompt. *(P.S.: We have to set or change the PATH system variable for Java if it is not done yet in your computer.)*

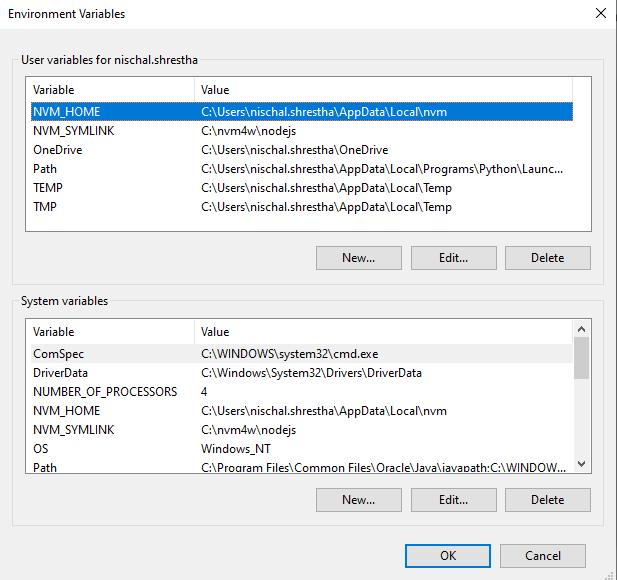
*The PATH is the system variable that our operating system uses to locate needed executables from the command line or Terminal window. The PATH system variable can be set using* ***System Utility*** *in the control panel on Windows, or in our shell’s startup file on Linux and Solaris.*



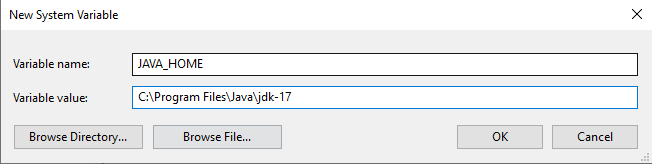
1. Setting up the environment variable.
   1. Firstly, we need to know where our JDK 17 was installed. In my case, the default location was “C:\Program Files\Java\jdk-17”
   2. Now, we have to access the environmental variable. We need to get to the System Environment Variables Dialog.



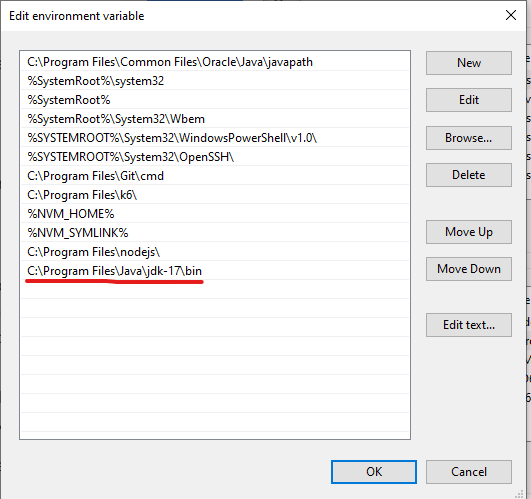
* 1. Then, in the “Environment Variables” dialog, I checked if “JAVA\_HOME” already existed.



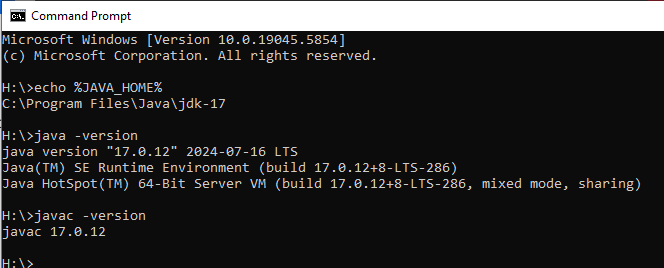
* 1. Configuring JAVA\_HOME



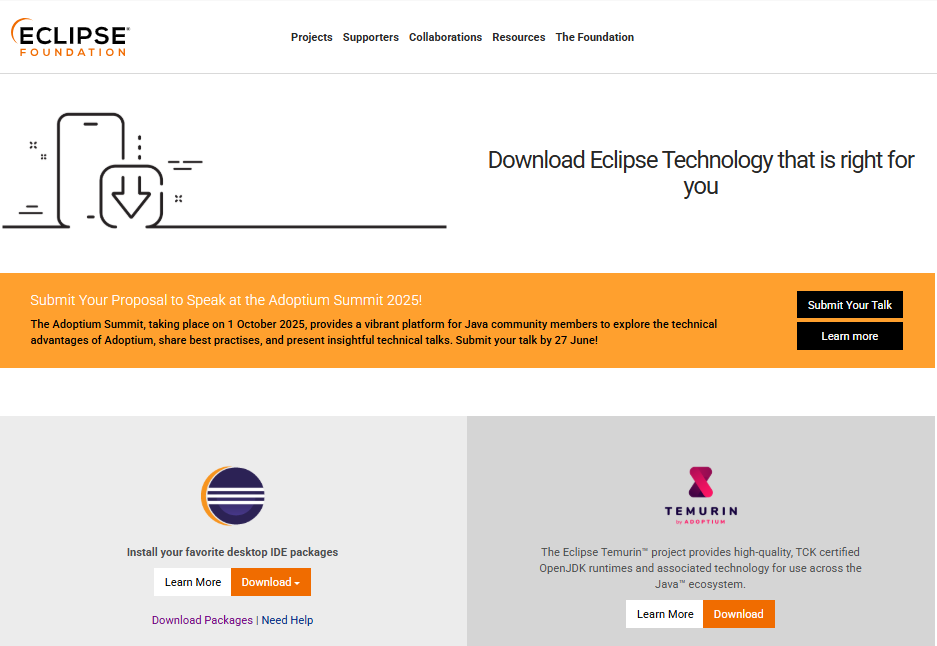
* 1. Updating the “Path” variable.



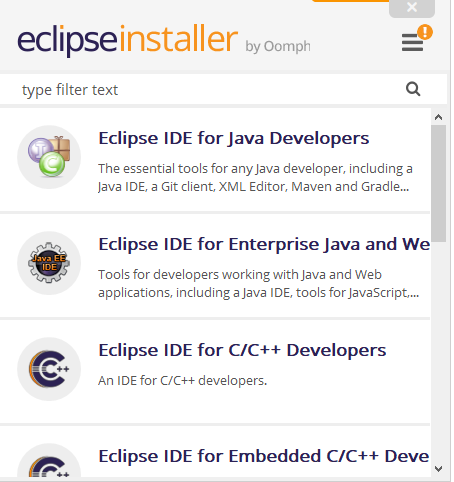
* 1. Verifying if everything worked

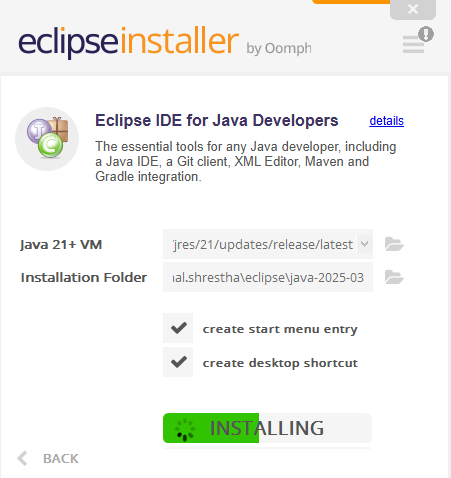


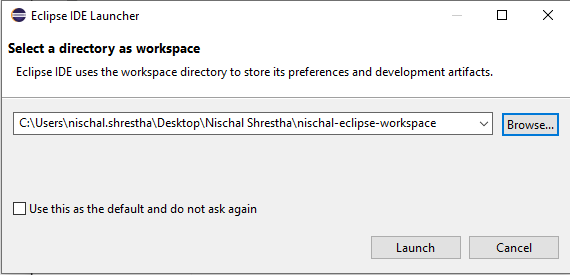
1. I installed Eclipse IDE on my system by downloading the latest version of Eclipse from “<https://www.eclipse.org/downloads/>”

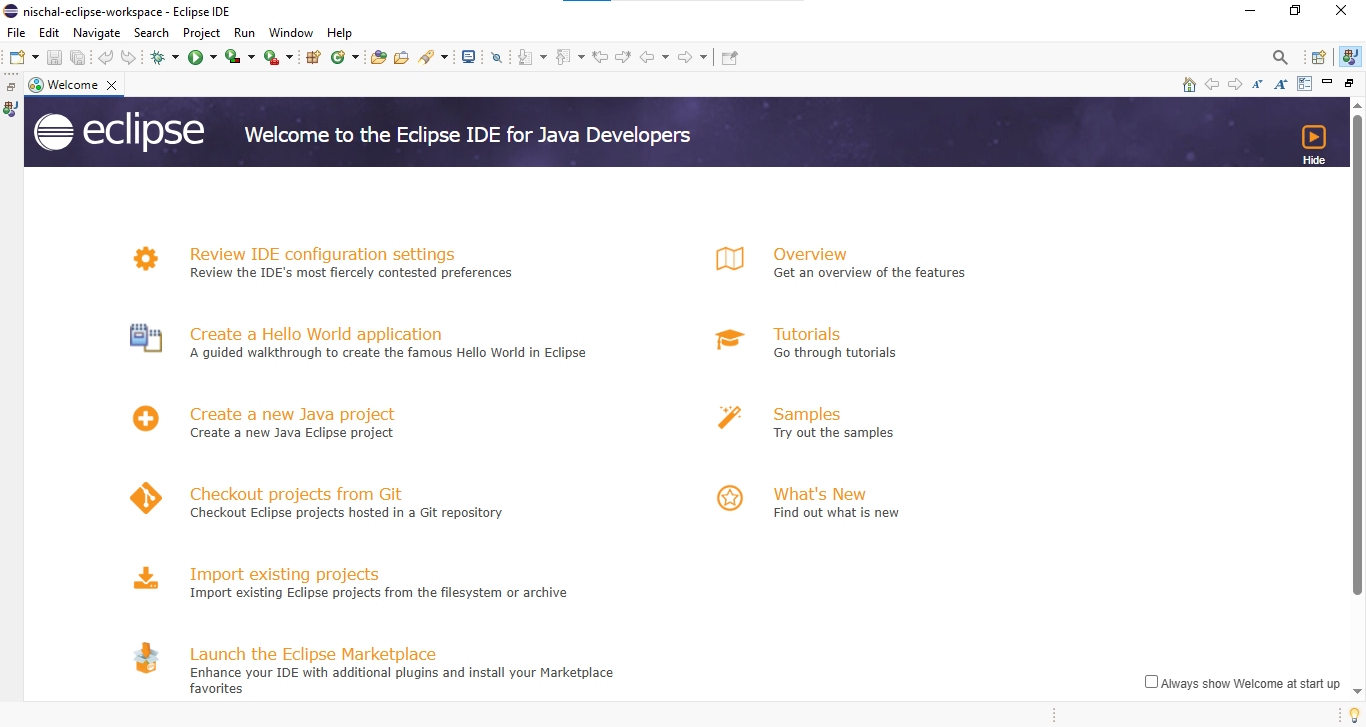


I chose Eclipse IDE for Java Developers for Selenium WebDriver.



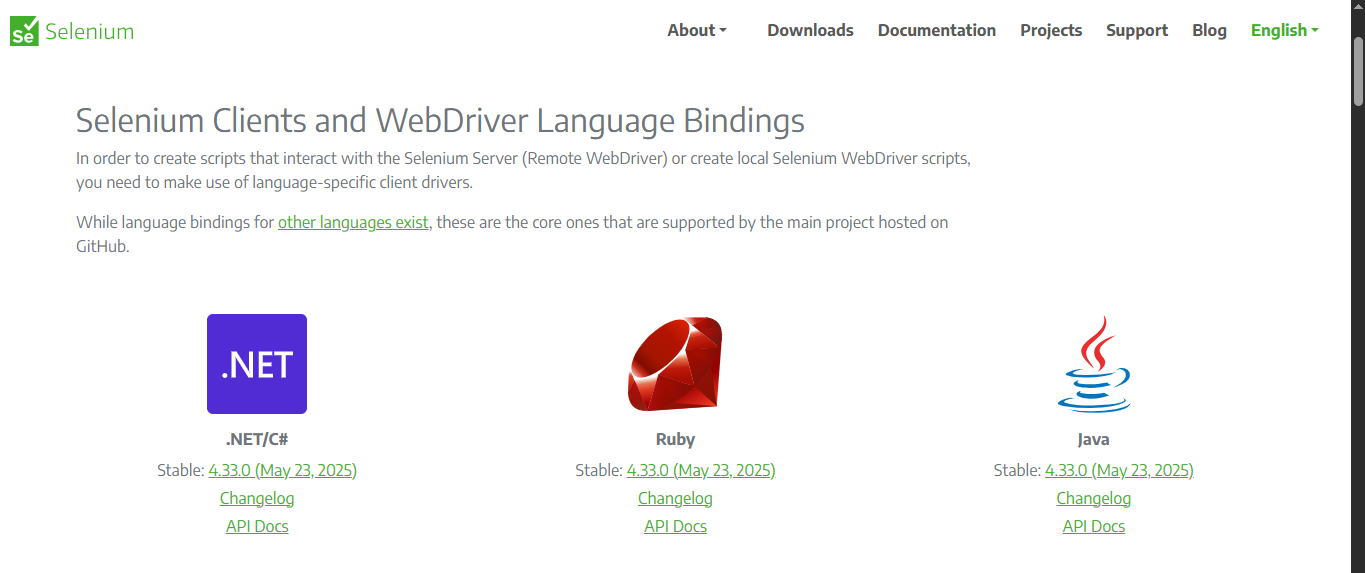




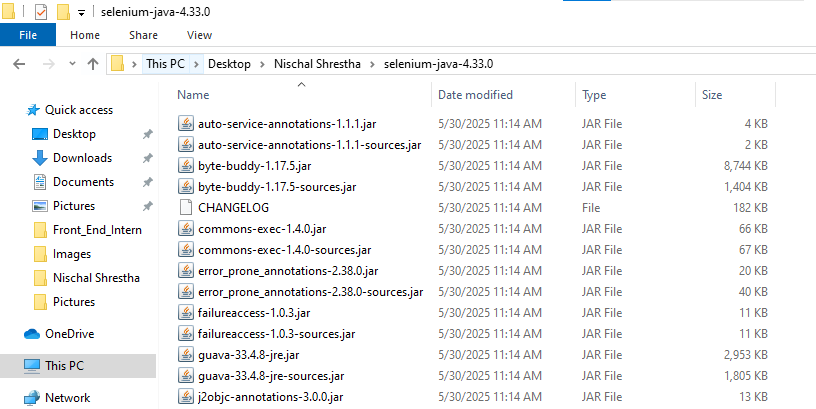


### Step 2: Downloading Selenium WebDriver Libraries

1. I installed Selenium WebDriver for Java on my system by downloading the latest version of Selenium WebDriver for Java from “<https://www.selenium.dev/downloads/>”

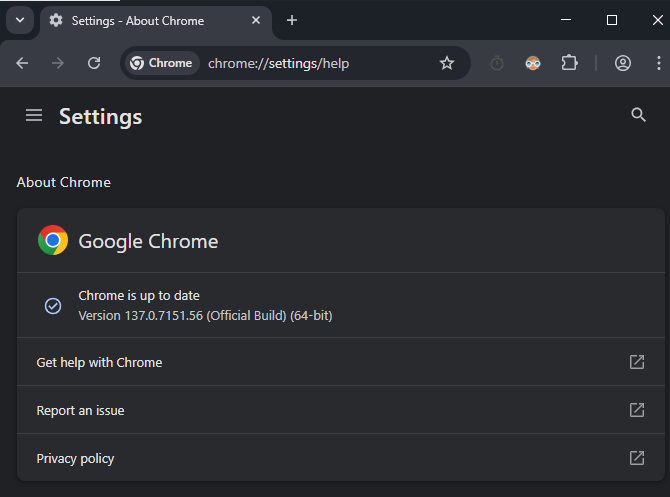


1. I extracted the ZIP file to the desired folder.

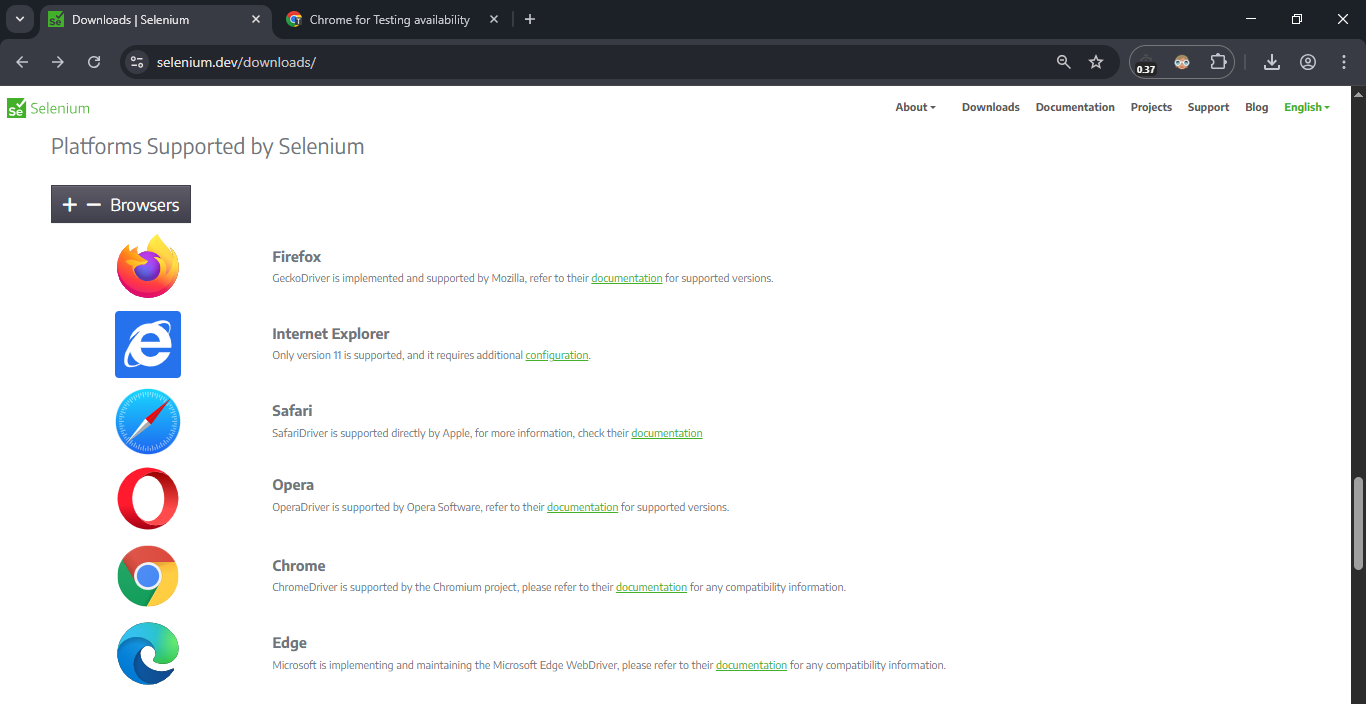


### Step 3: Download Browser Driver *(e.g., Chrome)*

1. Checking the browser version.



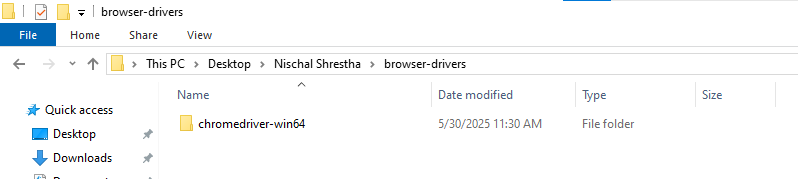
1. Downloading Browser Driver from “<https://www.selenium.dev/downloads/>” Browsers section *(for demo I downloaded Google Browser Driver)*

**

1. Depending on our **Browser version** that we have on our machine, we have to download that stable **Browser Driver.**

****

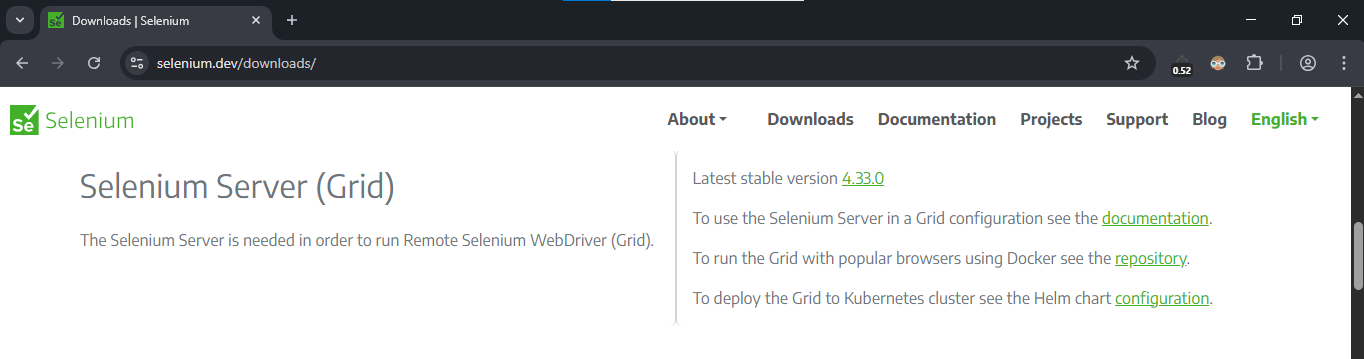
1. The downloaded zip file was extracted and its contents were moved to **'browserdrivers'**, a folder where all browser drivers are kept for easier organization.



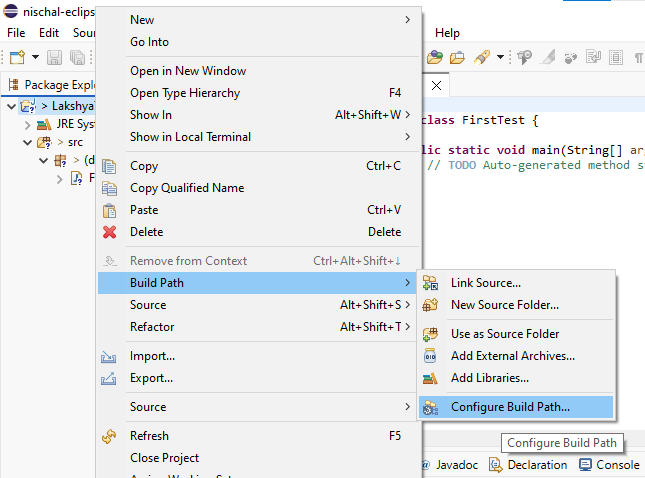
*[P.S.: We must set up our browser driver in our Java Project code so it can recognize which browser the script will launch.]*

### Step 4: Downloading the Selenium WebDriver jar

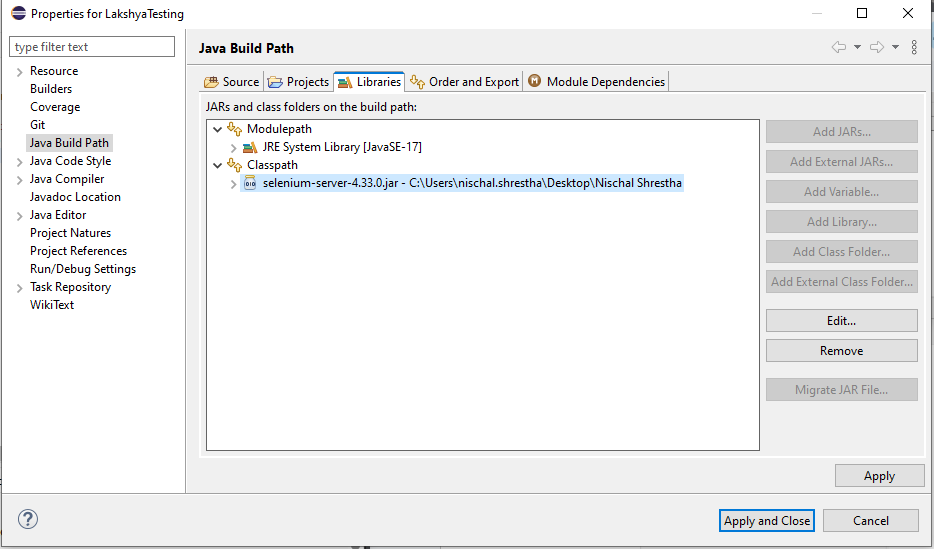
1. I downloaded Selenium WebDriver jar from “<https://www.selenium.dev/downloads/>”



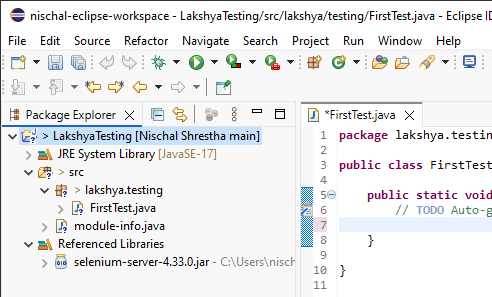
1. Importing the jar file in my project
2. Right click project > Build Path > Configure Build Path



1. Libraries > Add External JARs > File Path to the downloaded Selenium WebDriver JAR >> Open

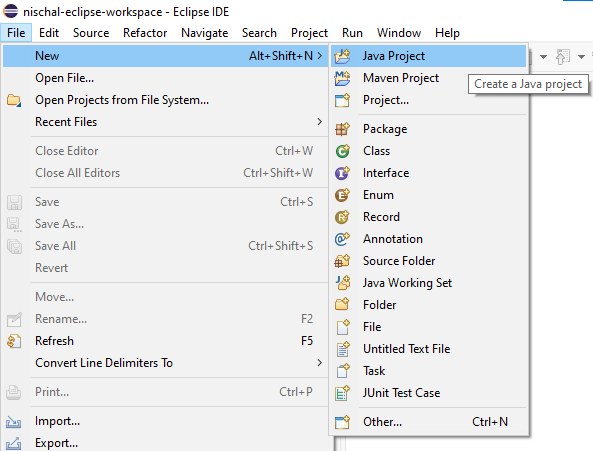


1. Apply and Close

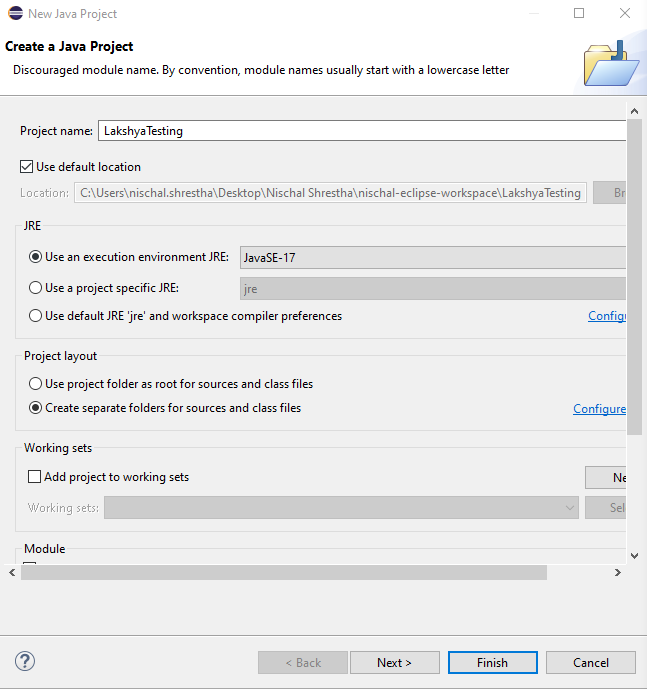


### Step 5: Creating Java Project

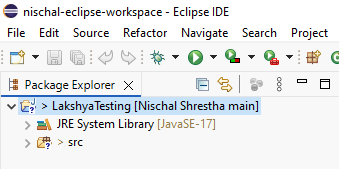
1. File > New > Java Project.



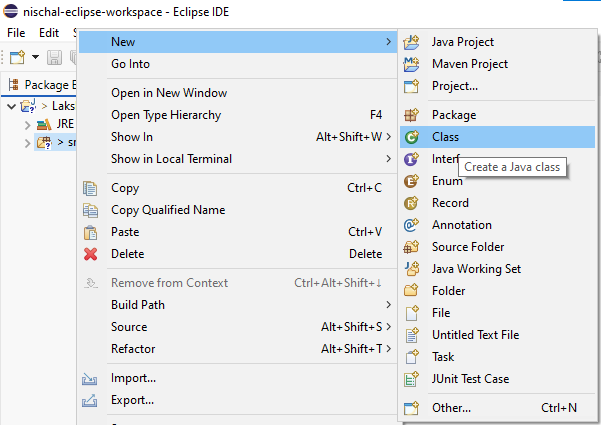
1. Then, I entered a name for my project “LakshyaTesting”. I also checked the execution environment JRE.

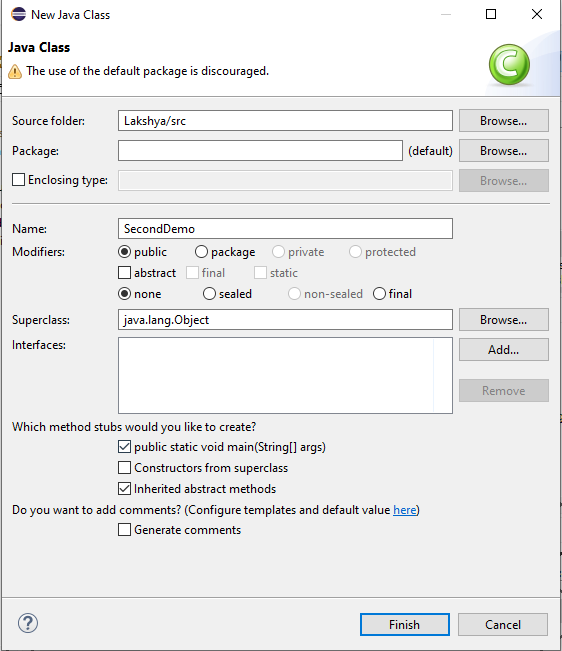


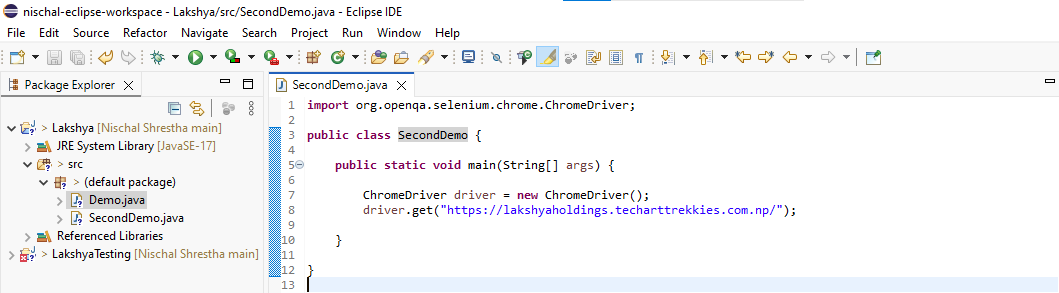
*[P.S.: The JRE system library is by default automatically imported so all the JAR files are already imported that are required for the Java project]*

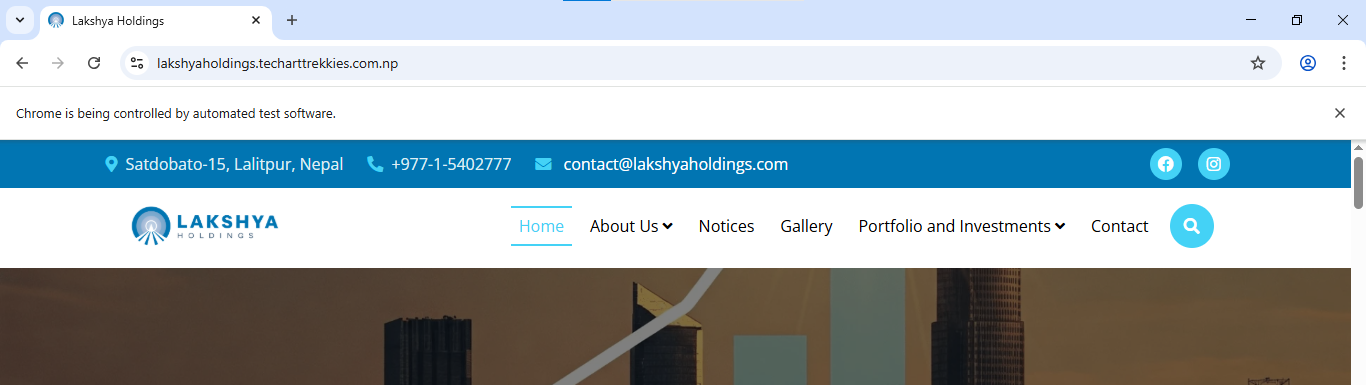


1. Create a Class File









# CONCLUSION

In conclusion, Selenium stands as an indispensable and highly versatile tool in the landscape of modern web automation. Its open-source nature, coupled with robust cross-browser, cross-language, and cross-platform capabilities, makes it an accessible and powerful solution for a diverse range of users, from software testers and developers to DevOps engineers and educational institutions. Its strong community support further ensures its ongoing relevance and adaptability, solidifying Selenium’s position as a cornerstone technology for efficient and effective web automation.