Department of Computer Science
American International University-Bangladesh

Course Name: SOFTWARE QUALITY AND TESTING

"Assignment on Automated Testing Tool"

Section: A

Supervised By:

ABHIJIT BHOWMIK

Associate Professor & Special Assistant [OSA], Computer Science

Submitted By:

Soily Ghosh Sneha

ID: 20-43702-2

Submission Date: August 8, 2023.

Selenium Testing Procedure:

Step 1- Create project in IntelliJ IDEA

First, we should open the IntelliJ IDEA IDE and create a new Java project. We should first create a package in the src folder of that Java project, and then create a public class inside that package.

Step 2- Configure pot and add dependencies

Configure the build path for the generated Java project. To do so, after adding the selenium external jar file to the module path in the libraries section, we must apply. The selenium external jar file will be listed in the referred libraries section after it has been used.

Step 3- Website automated testing steps

Steps	Step Description
1	Go to "http://localhost/RMS/
2	Click Sign Up Link from upper right corner
3	Enter user name: ‘malik’ in user text field
4	Enter password:’4444’ in password field
5	Click Log In button
6	Check the portfolio

Step 4- DOM and FINDELEMENT

1. Navigation Website.
2. Click.
3. Type Test.
4. Portfolio navigation Test.

Step 5- Run and Test

For the purposes of website testing, we must run and execute the program code.

Selenium Installation Flowchart:

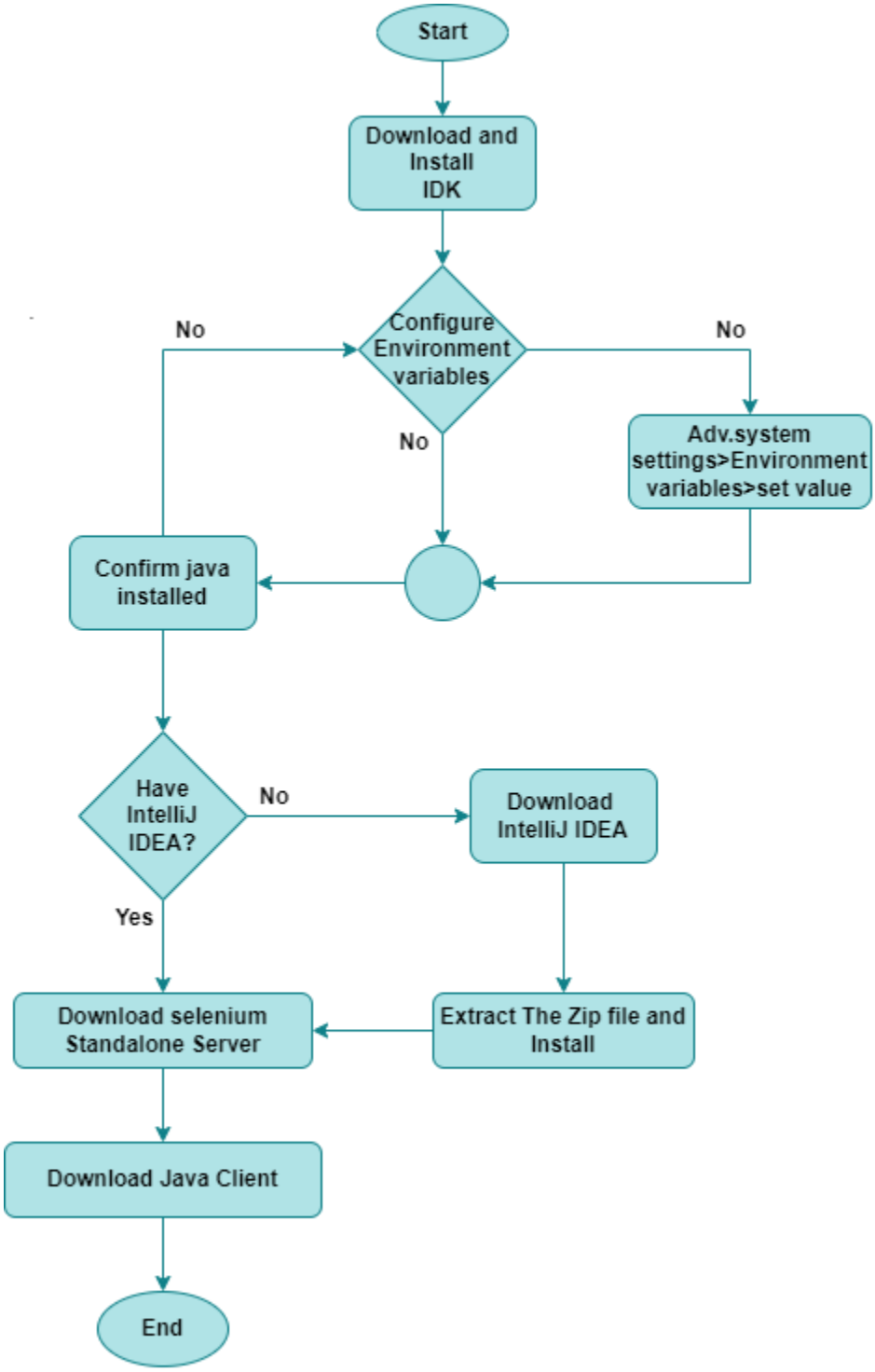

Figure 1: Flowchart of Selenium Installation.

Installation Process of Java for Windows:

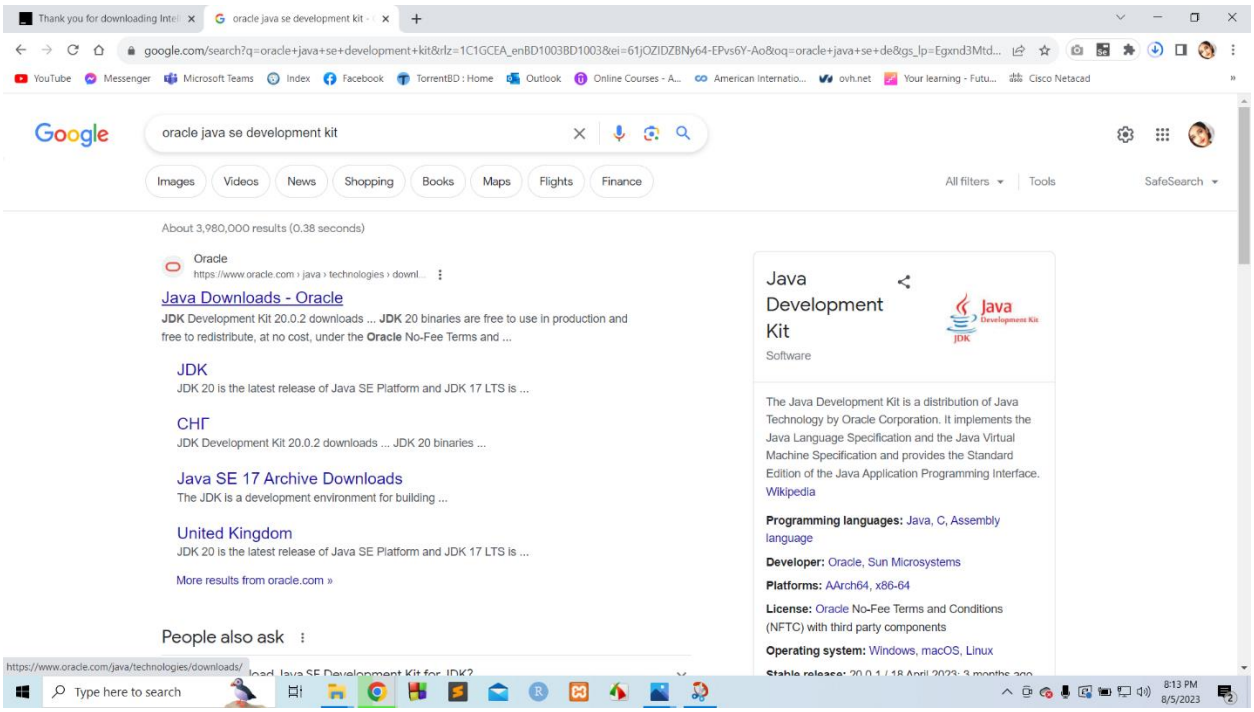

Figure 2: Go to the chrome to download Java.

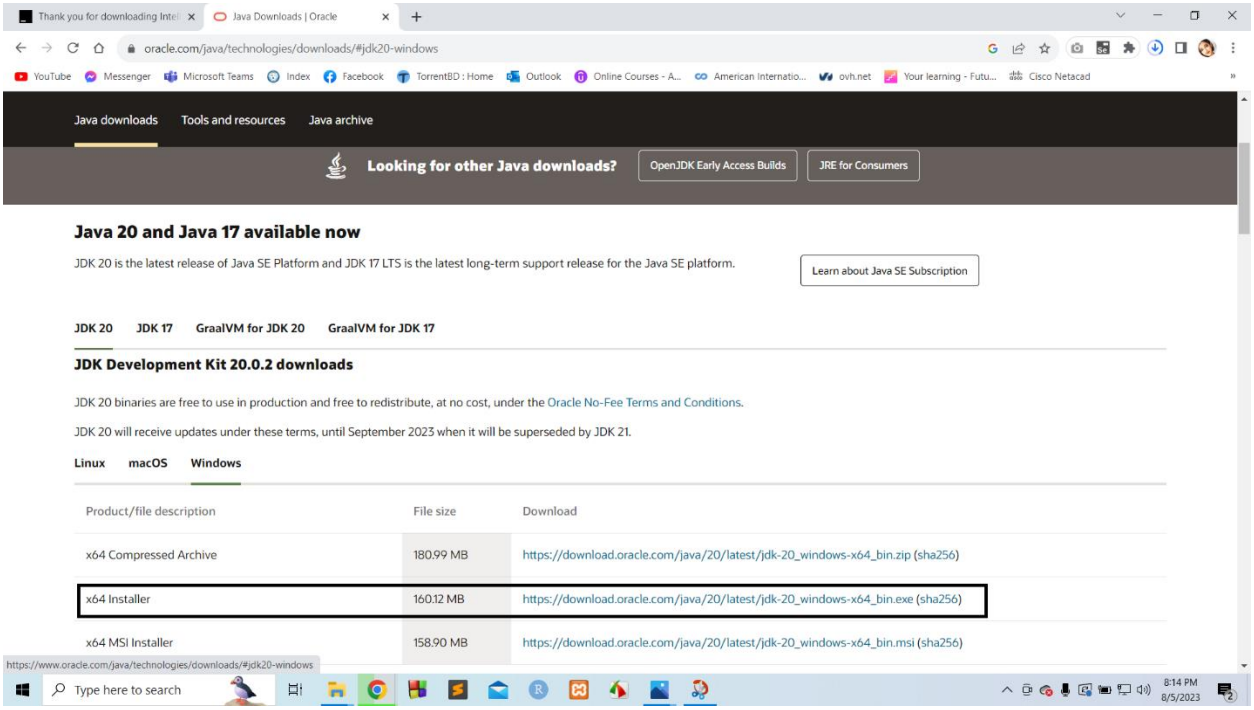

Figure 3: The link to download 64bits installer for window.

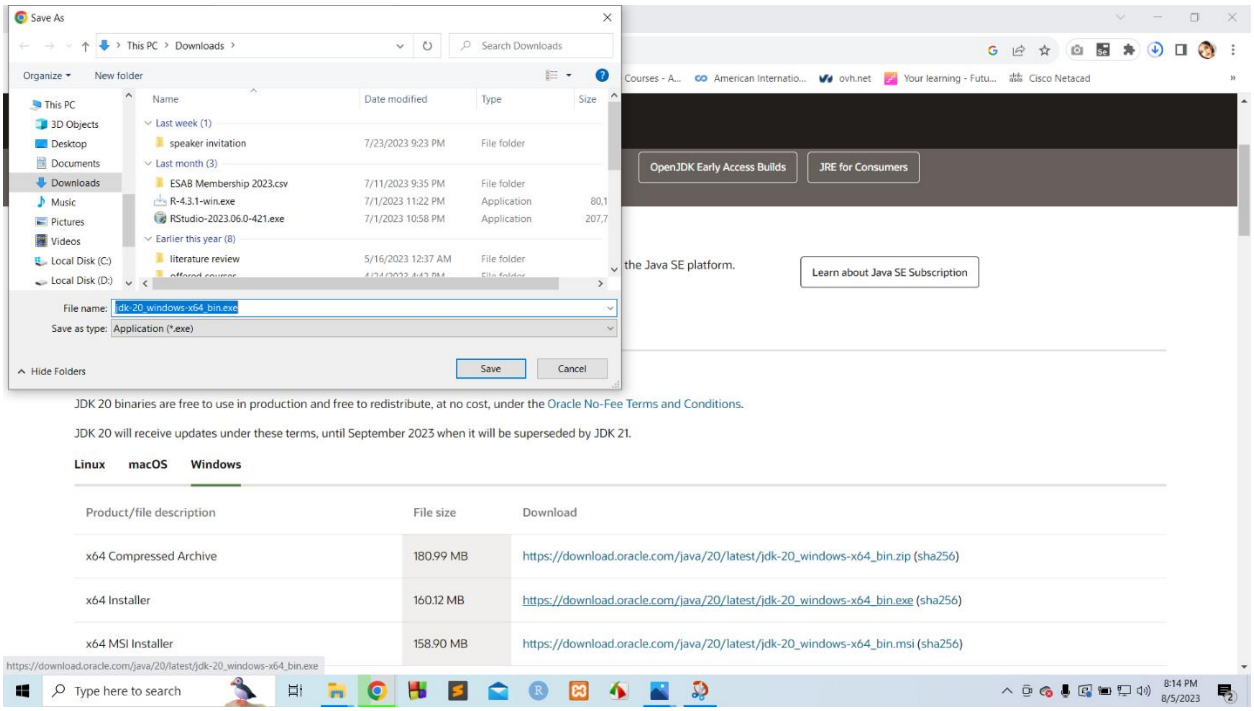

Figure 4: The download process.

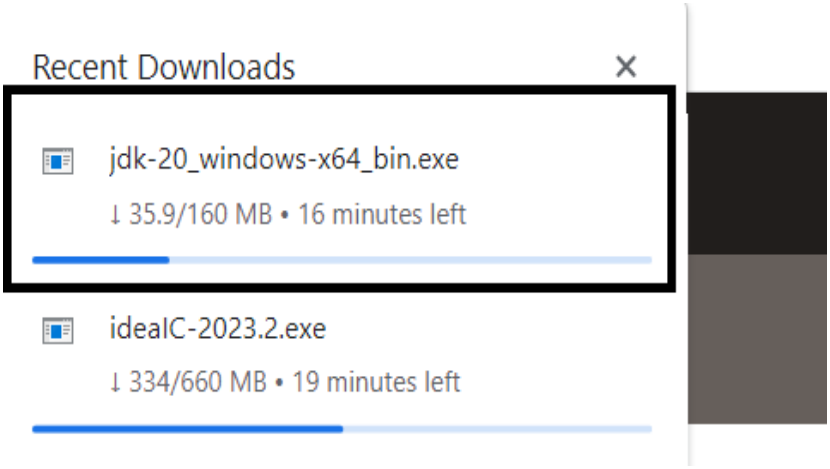

Figure 5: While downloading the Java.

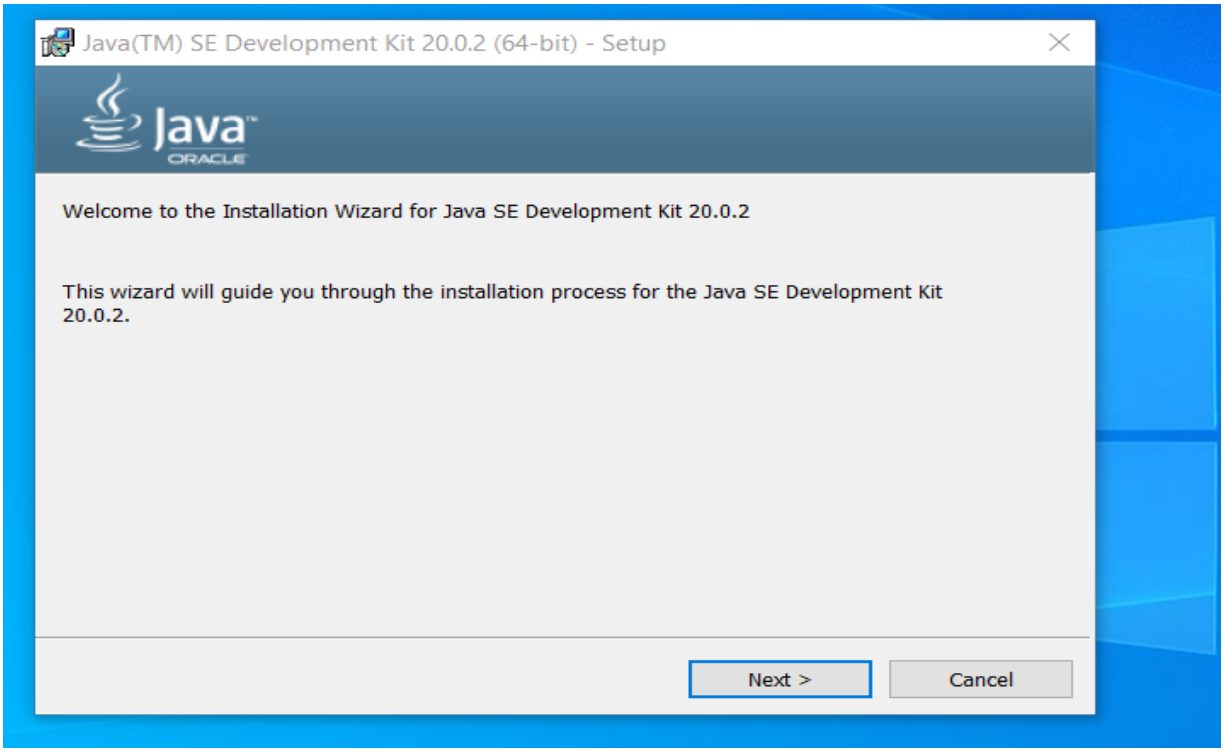

Figure 6: After downloading the installing start process.

Figure 7: The installing process.

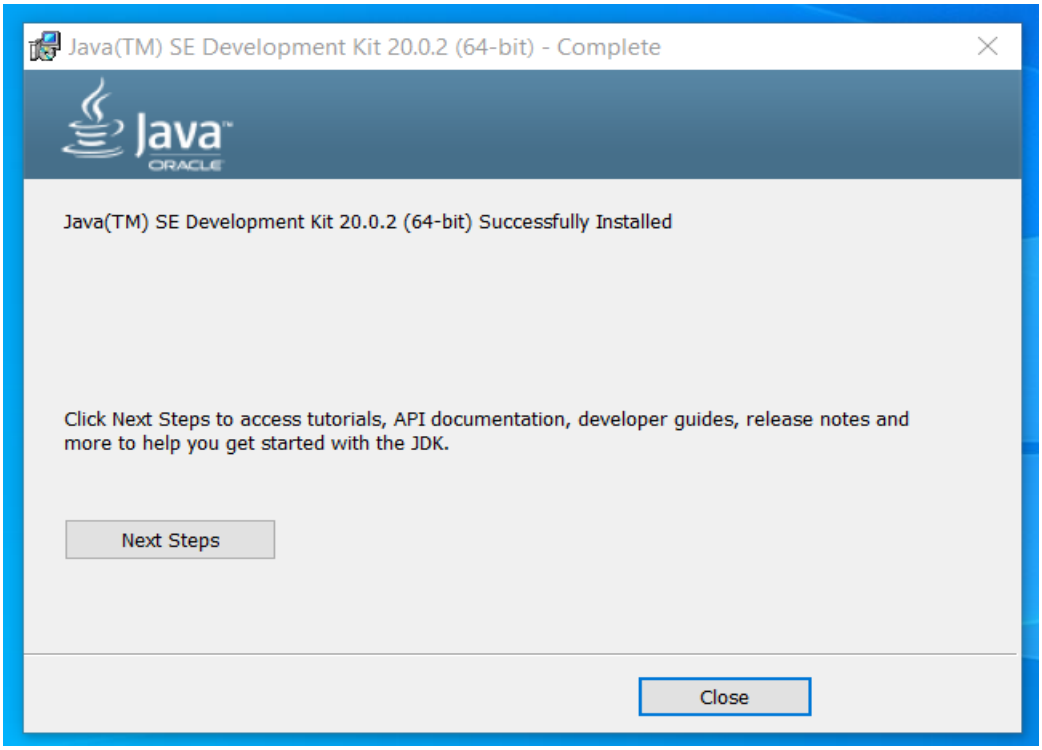

Figure 8: After the Successful Installation of Java.

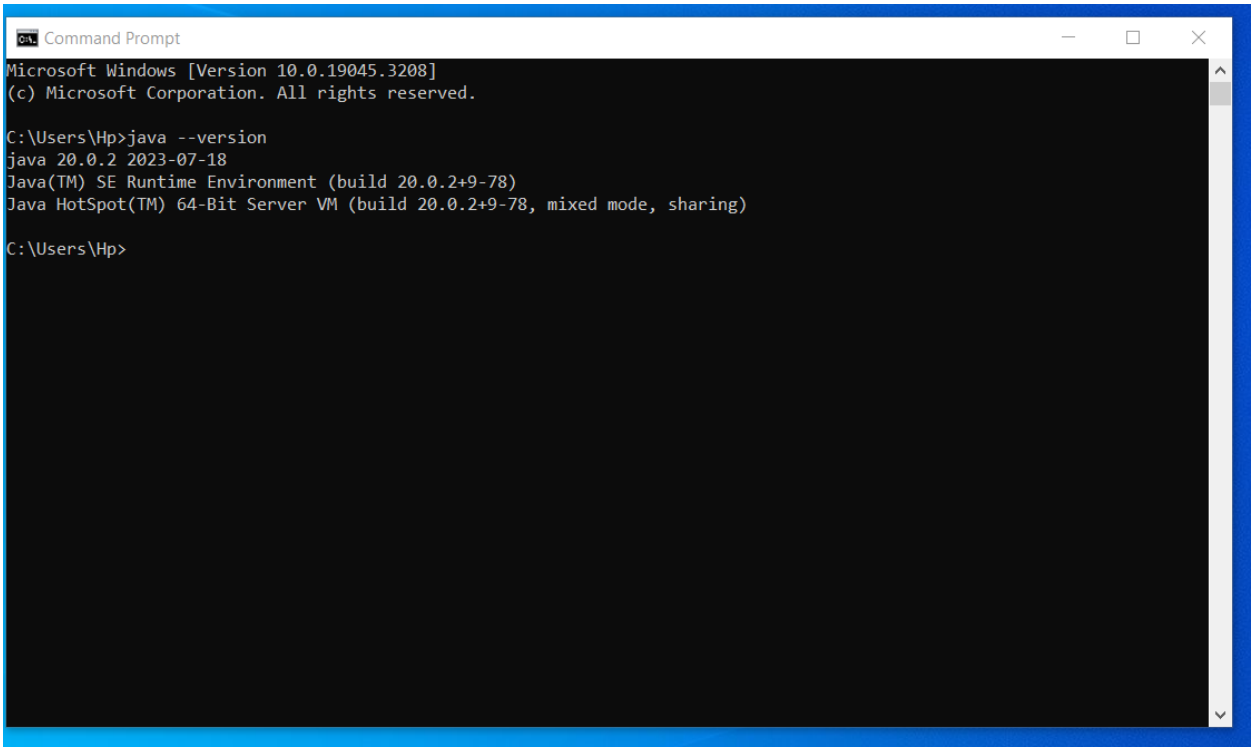

Figure 9: Checking the java version from the Command prompt.

Installation Process of IntelliJ IDEA:

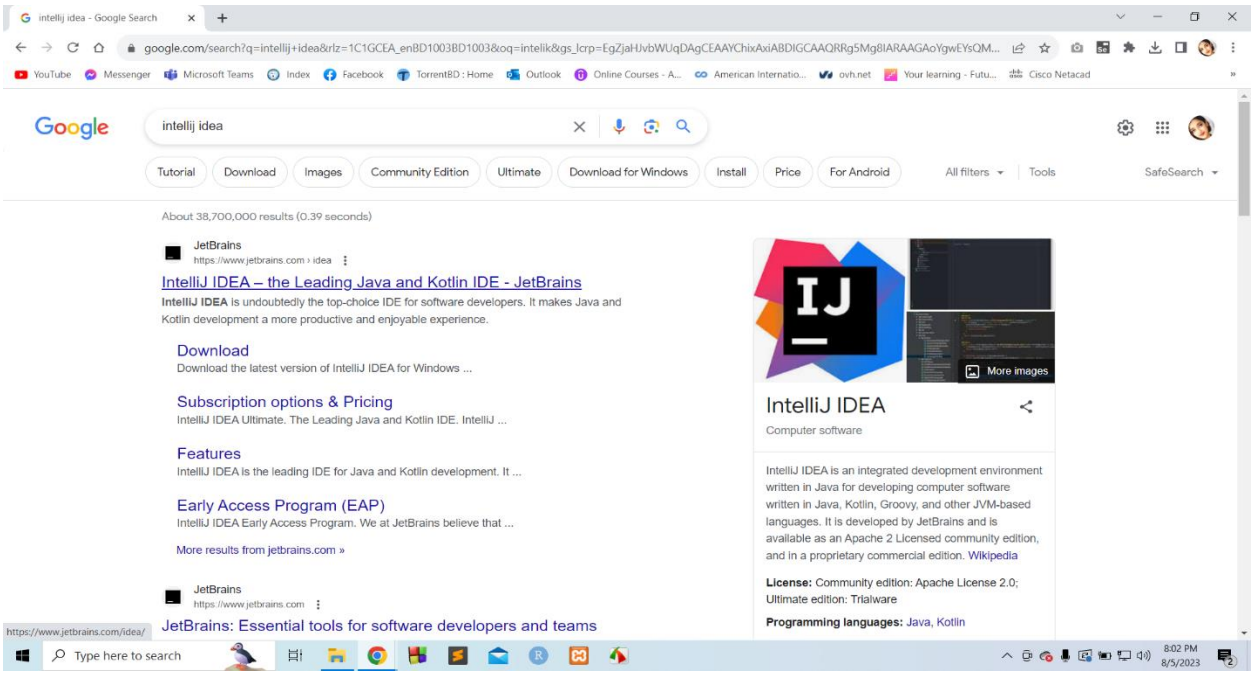

Figure 10: Go to the chrome to download IntelliJ IDEA.

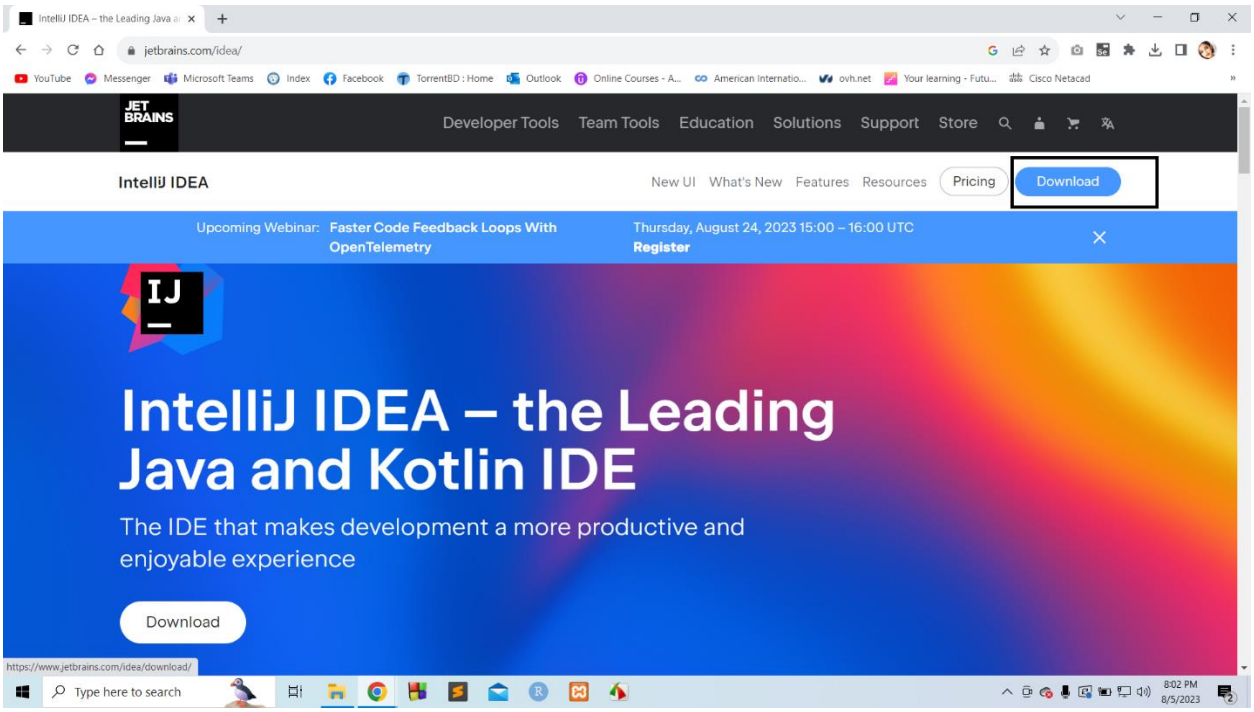

Figure 11: Downloading the IntelliJ IDEA for window.

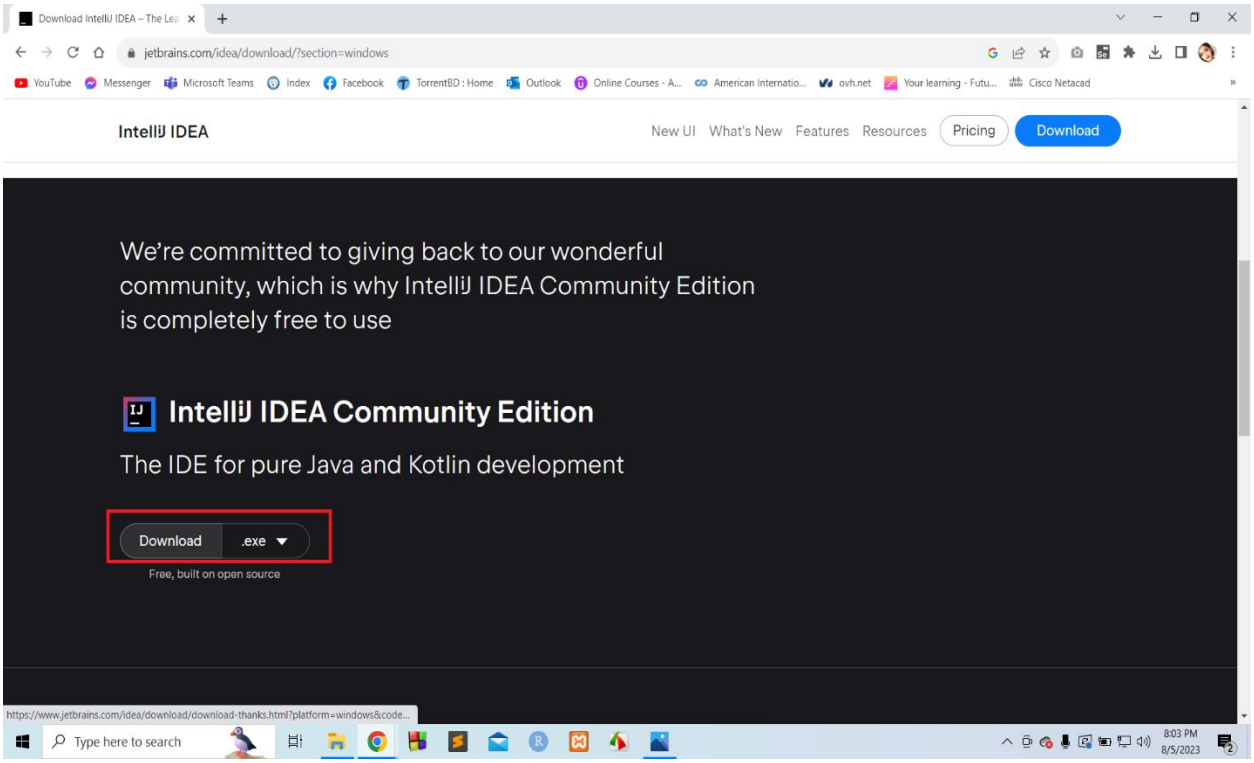

Figure 12: Downloading the IntelliJ IDEA Community Edition.

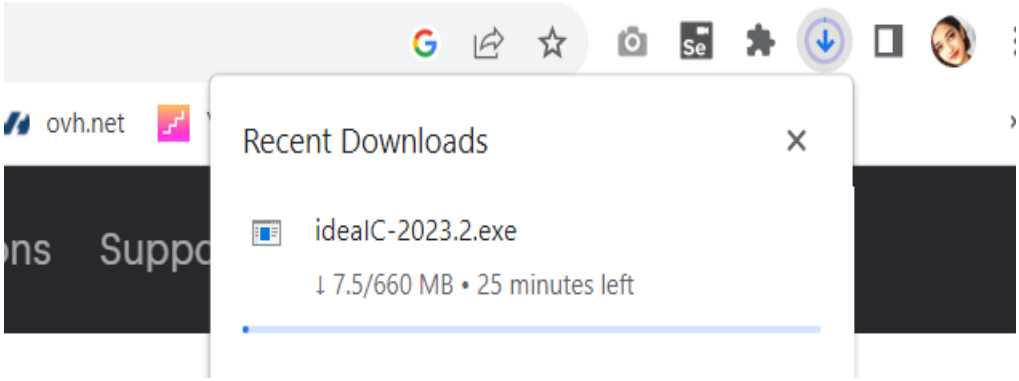

Figure 13: While downloading the IntelliJ IDEA.

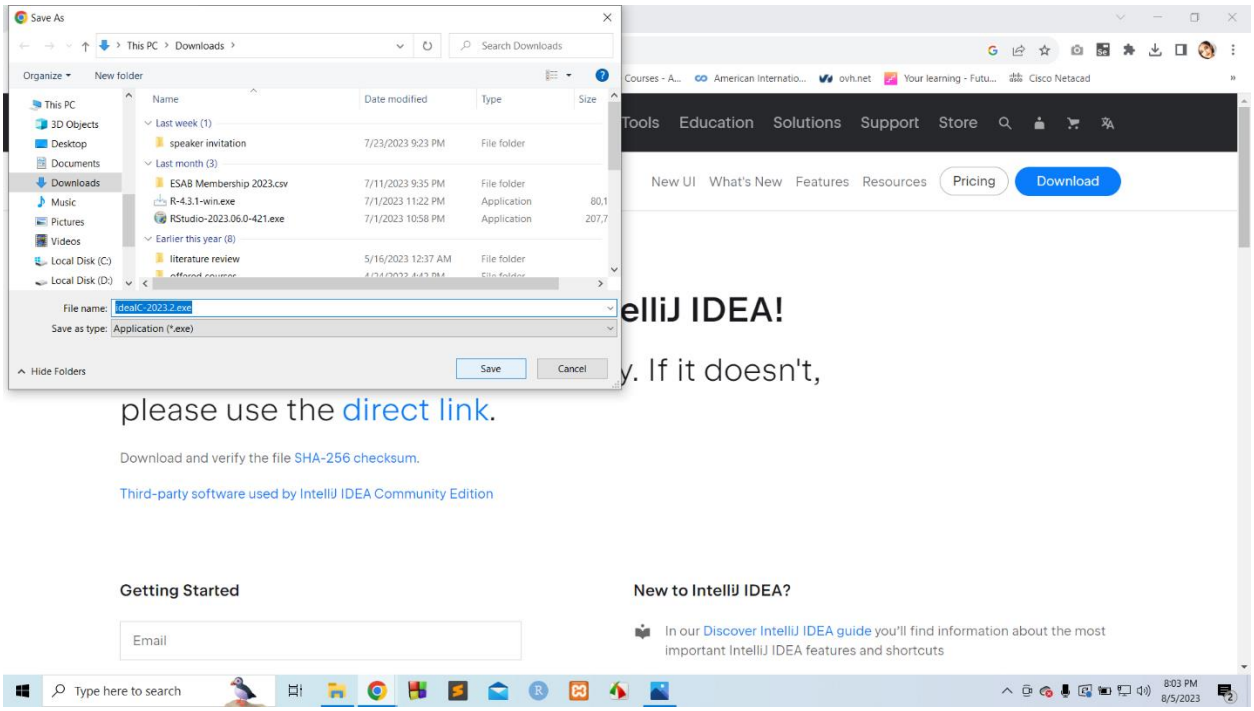

Figure 14: Downloading process of the IntelliJ IDEA.

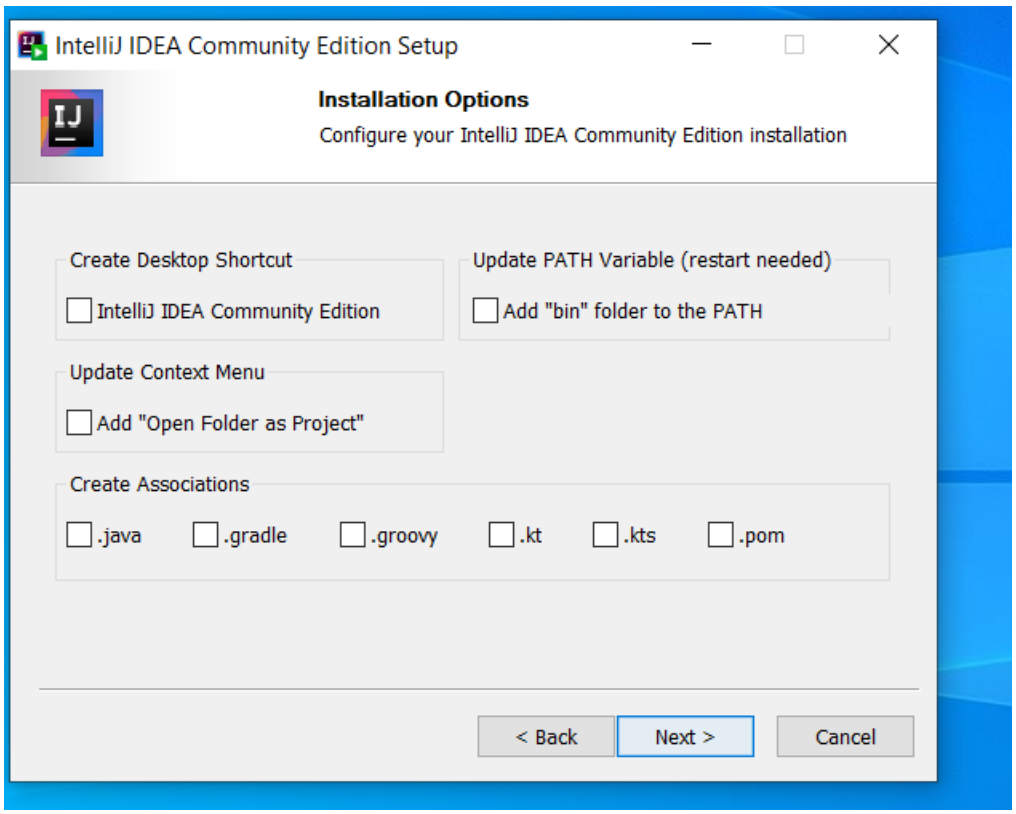

Figure 15: After download the start of installation.

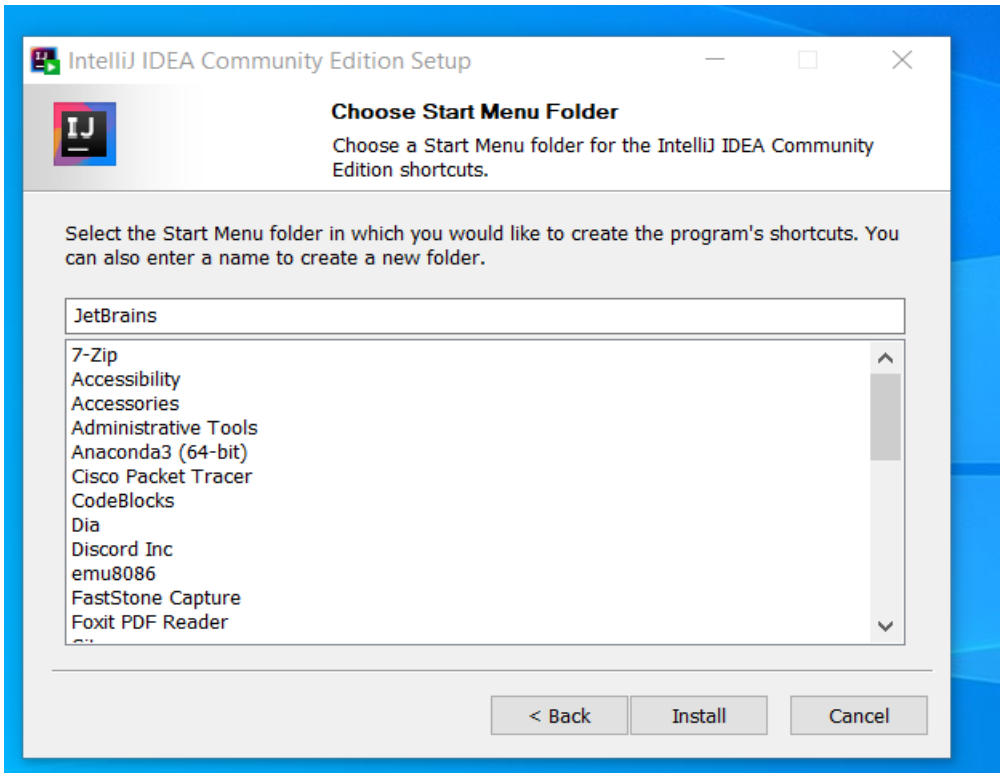

Figure 16: Choosing the menu folder of IntelliJ IDEA.

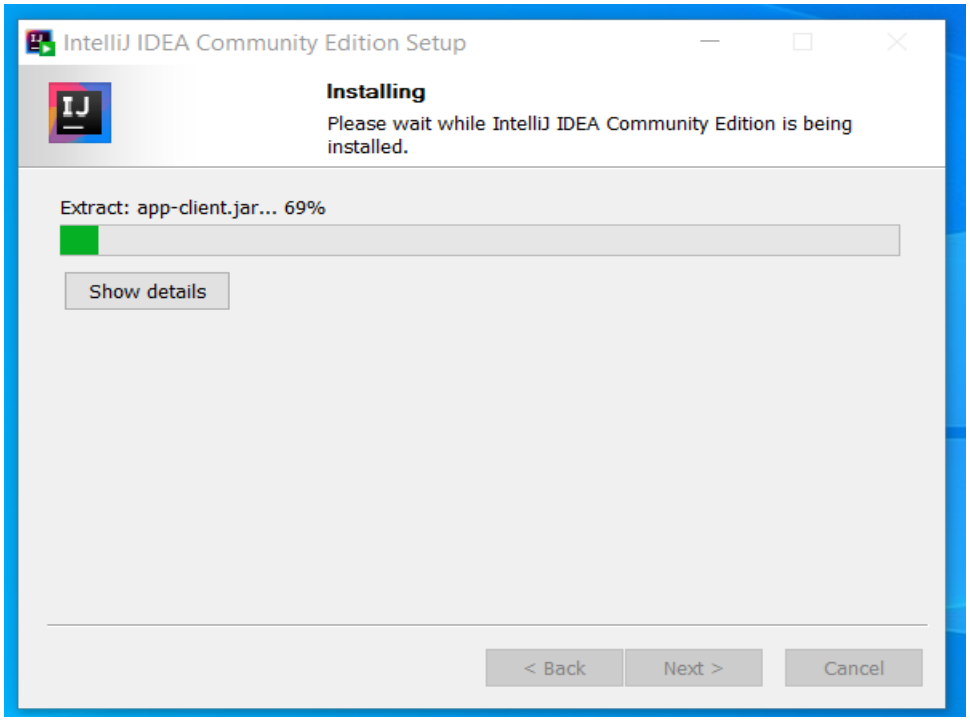

Figure 17: Installing percentage.

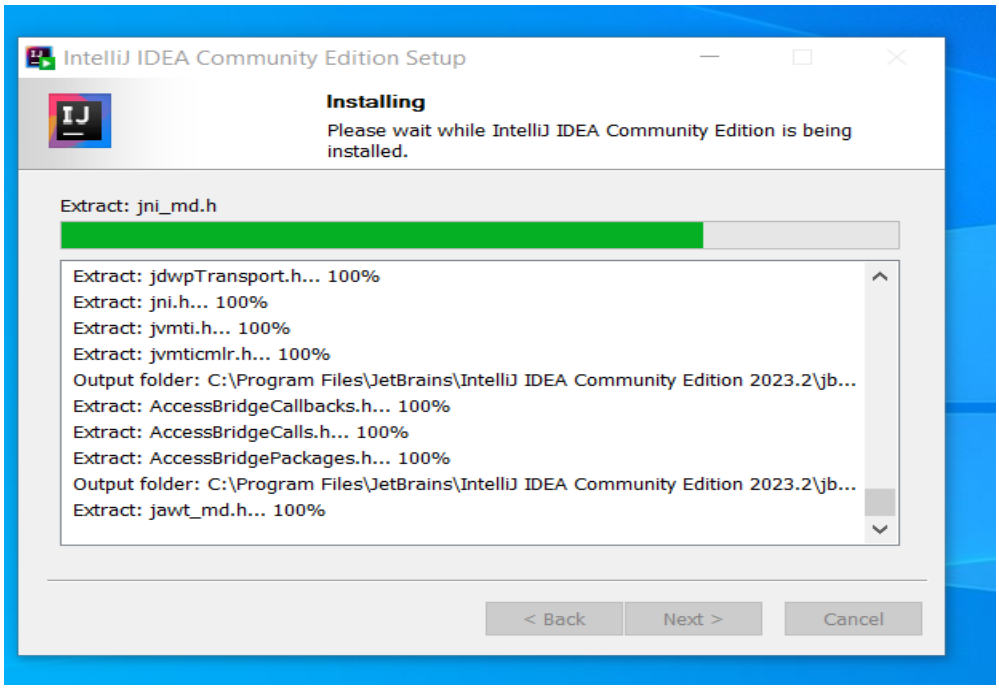

Figure 18: Installing percentage increasing.

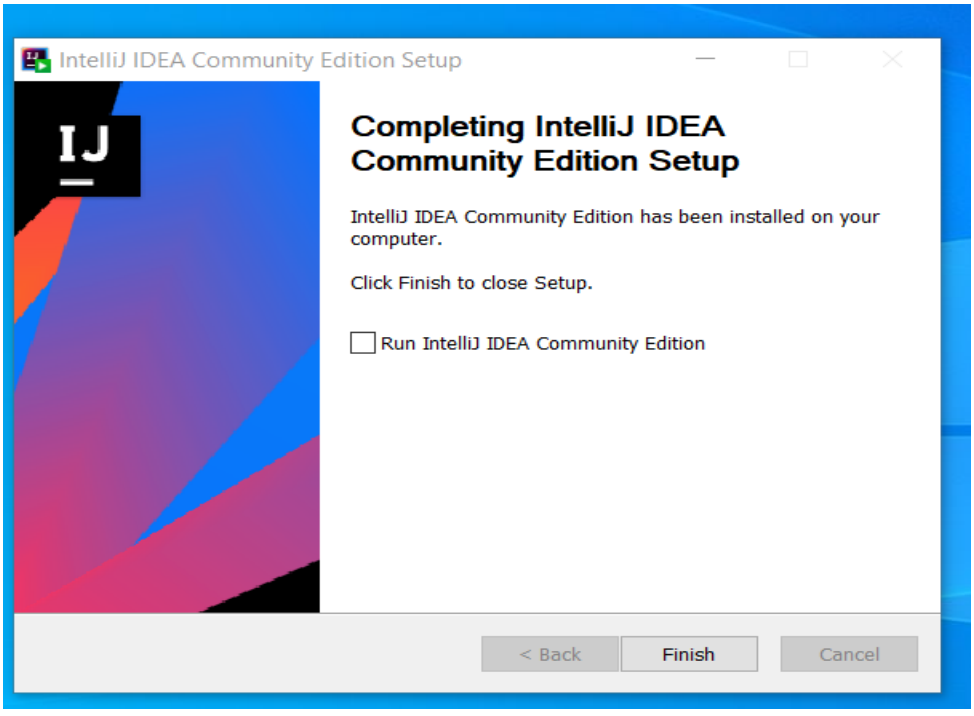

Figure 19: After successful installation of IntelliJ IDEA.

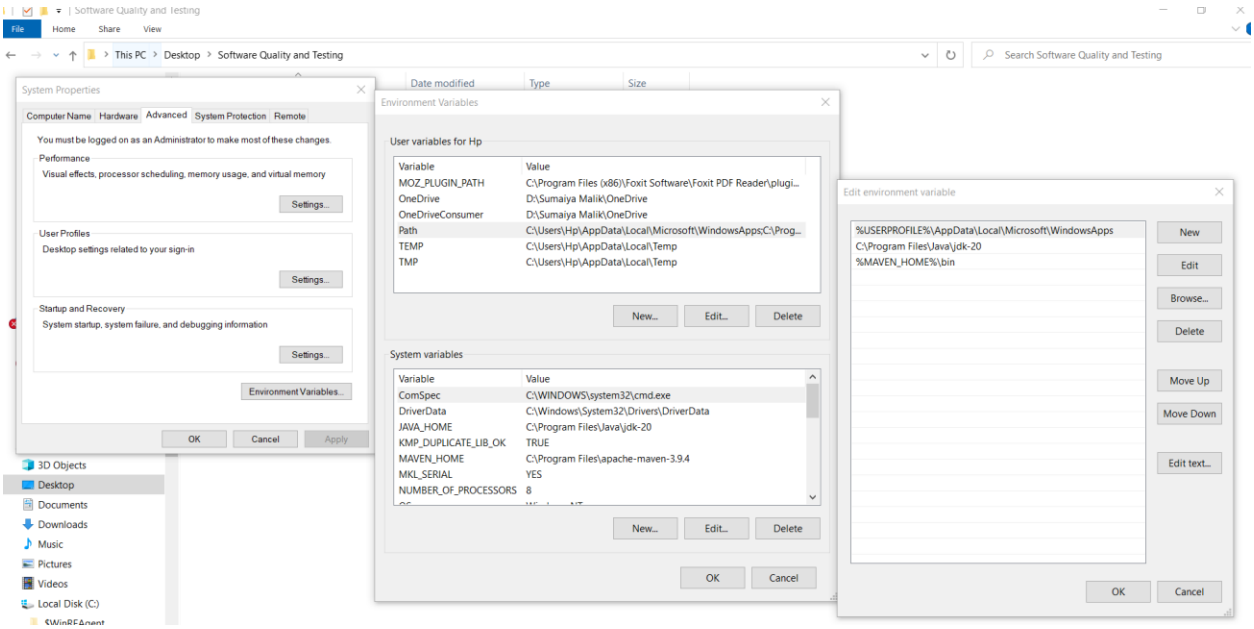

Figure 20: JDK-20 Environment setup.

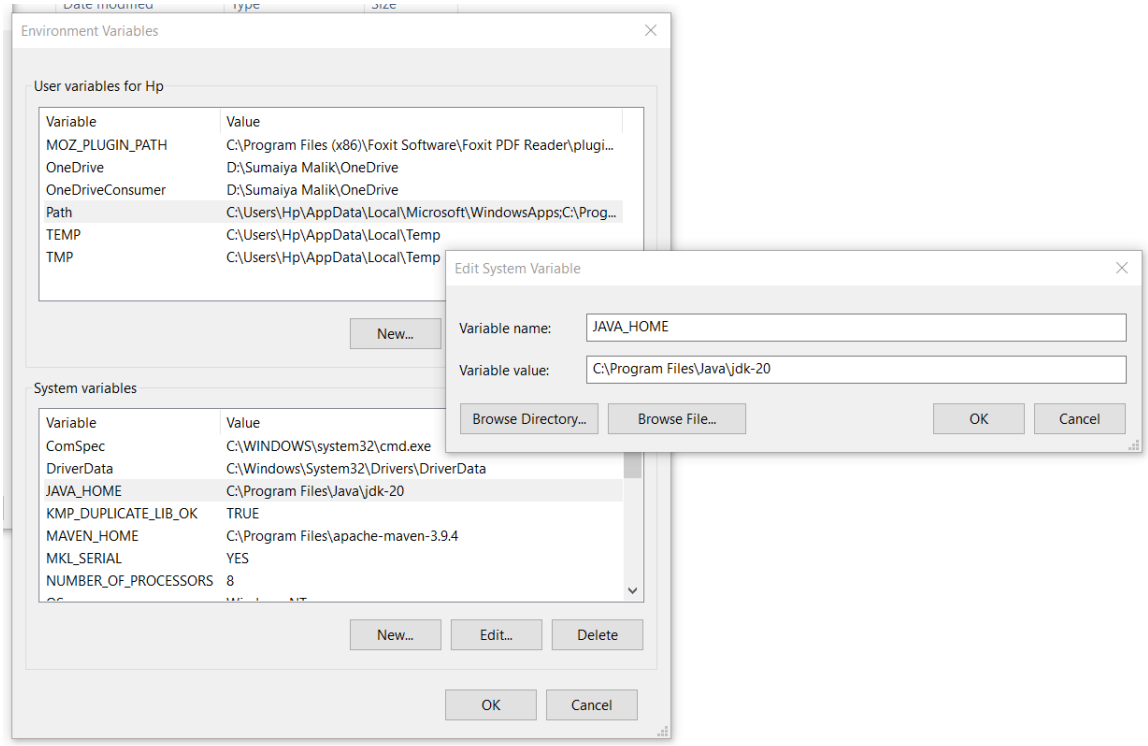

Figure 21: Jave Environment Variable Path.

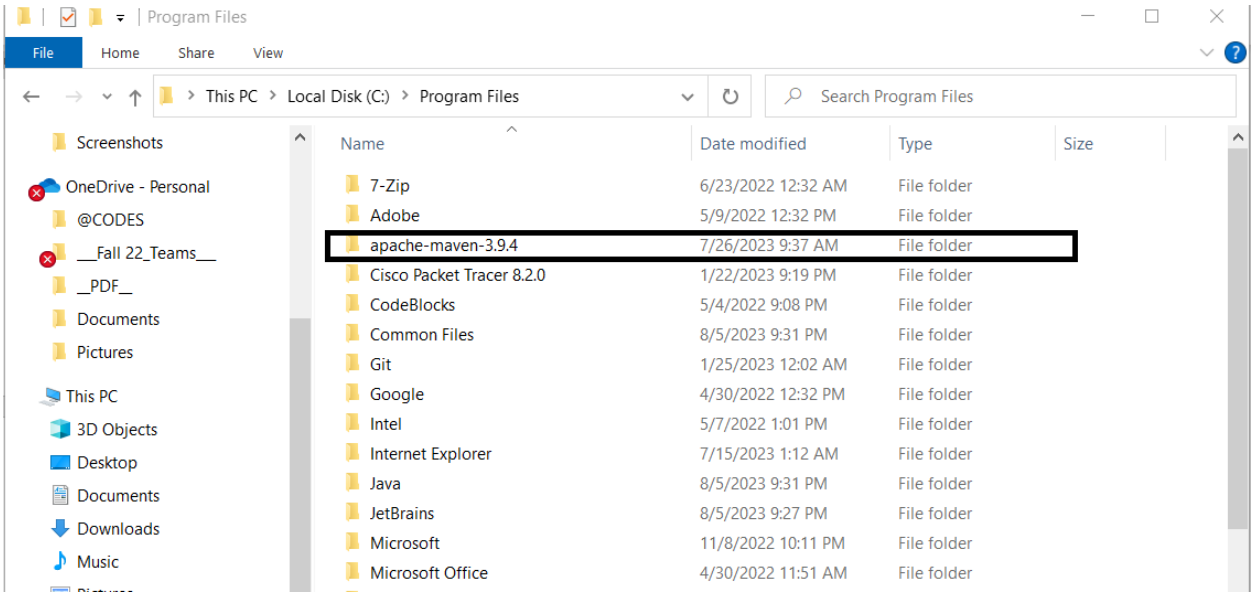

Figure 22: Downloaded Maven in C drive.

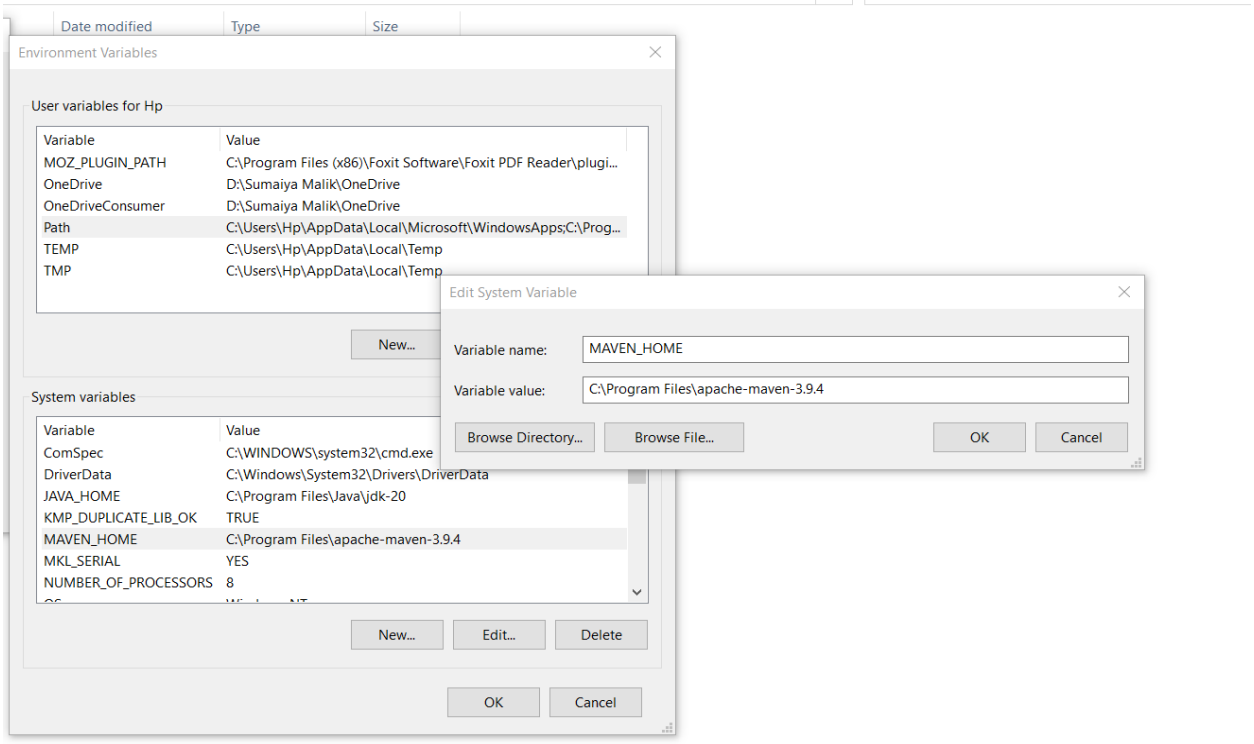

Figure 23: Maven Environment Setup.

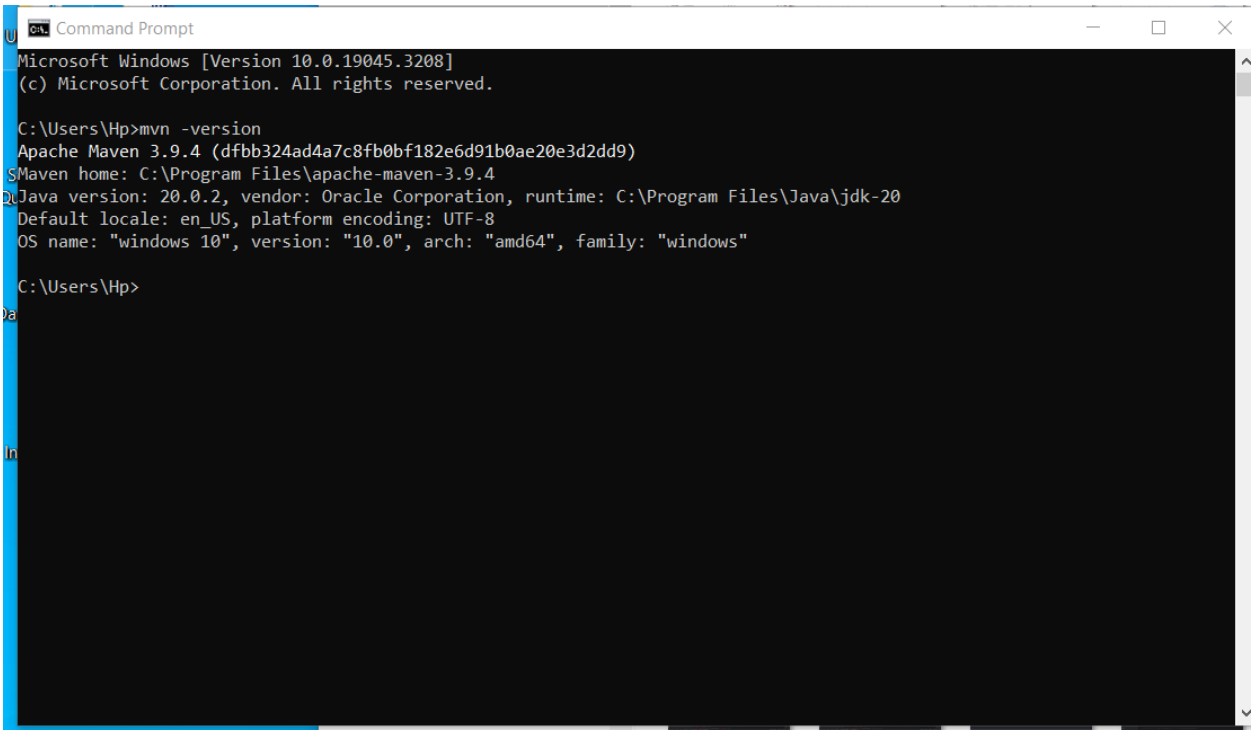

Figure 24: Checking the Maven version from Command Prompt.

Using the IntelliJ IDEA IDE:

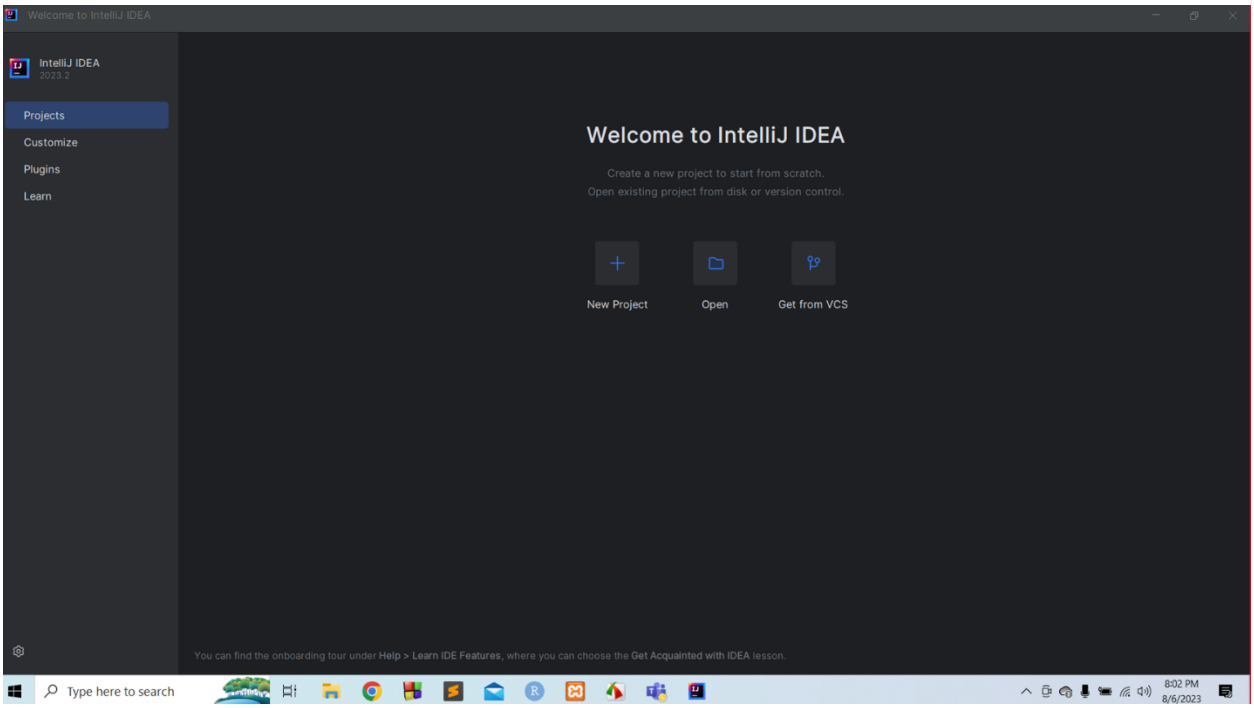

Figure 25: Opening the IntelliJ IDEA after installation.

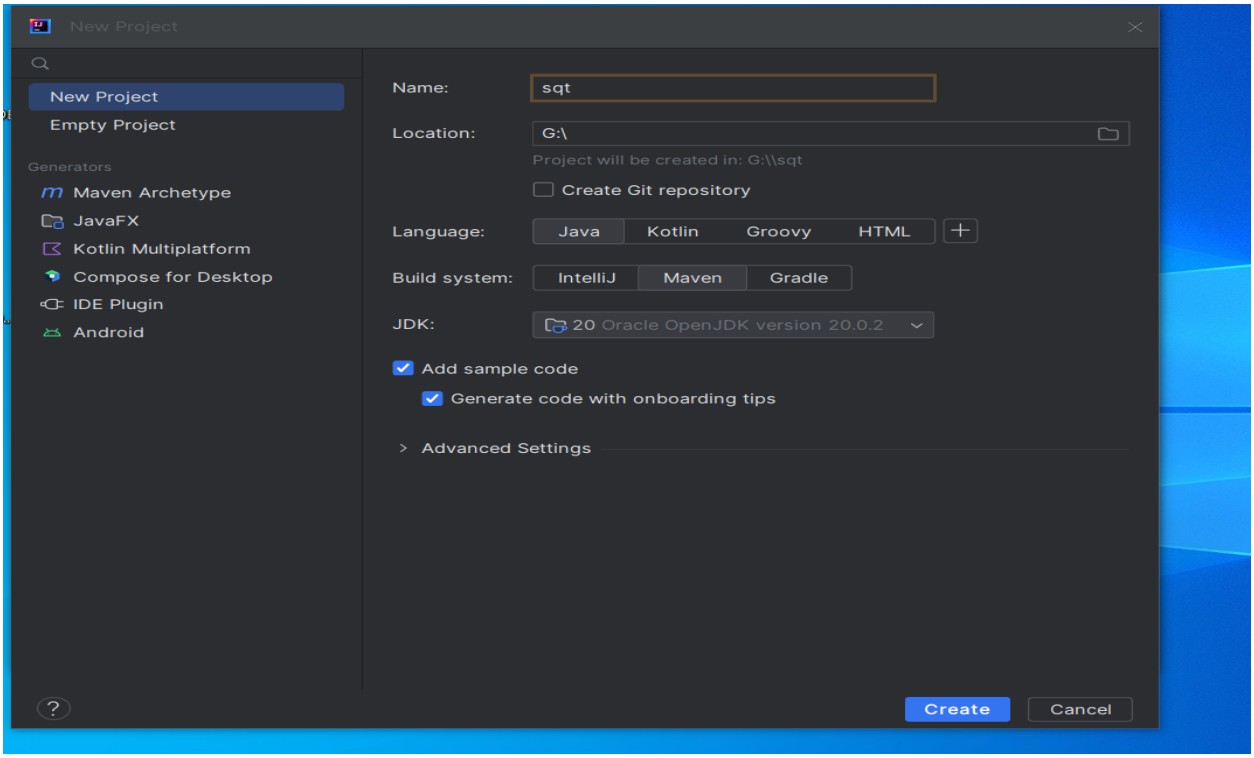

Figure 26: Creating a new project in IntelliJ IDEA IDE with sqt name folder.

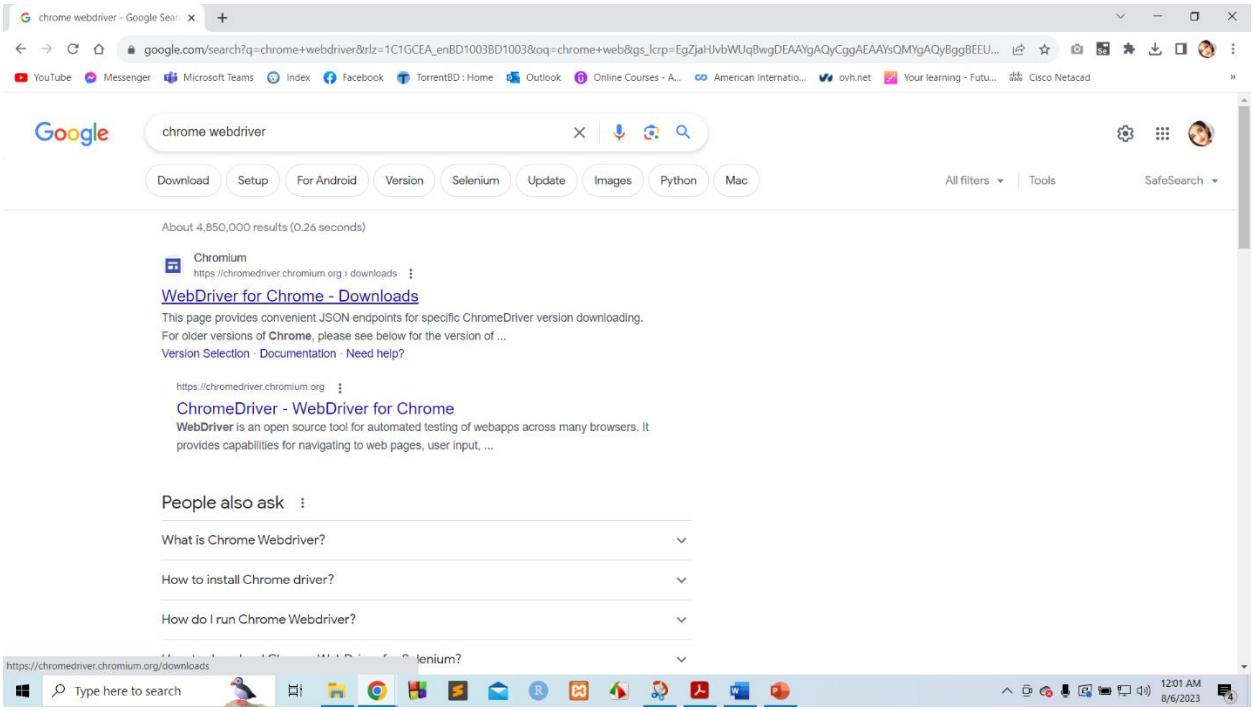

Figure 27: Downloading the WebDriver for Chrome.

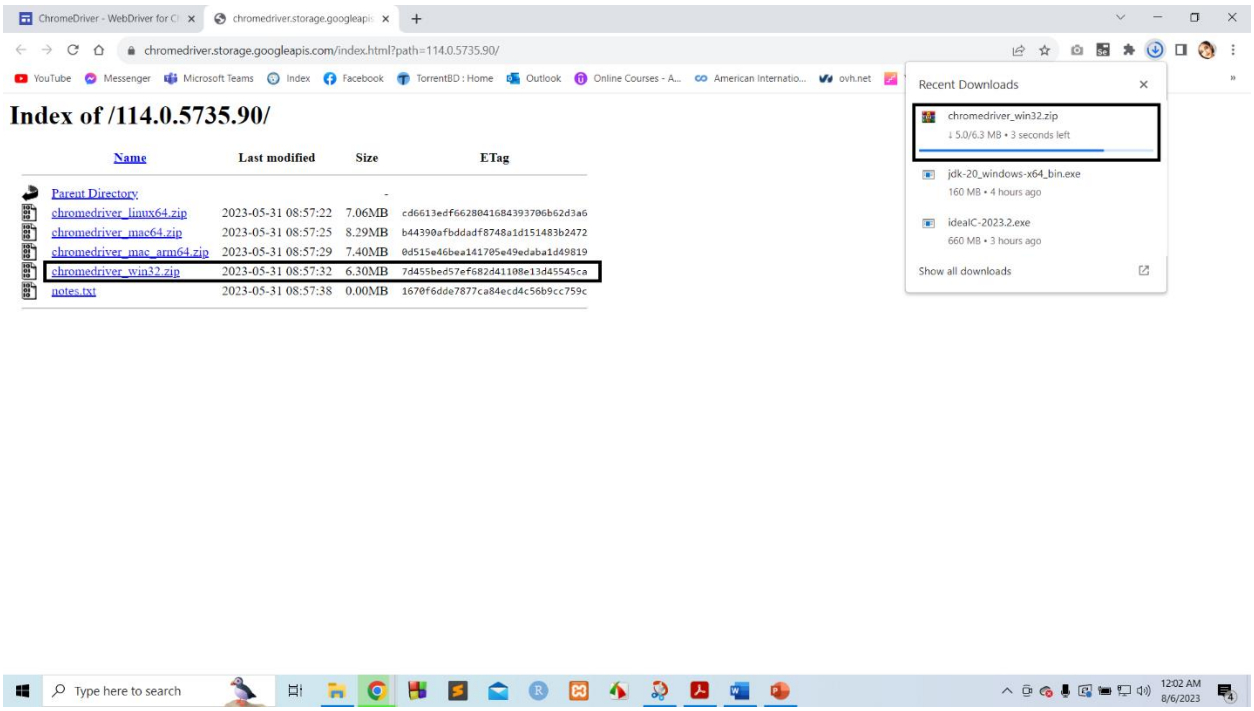

Figure 28: Downloading the WebDriver for Chrome of 32 bit-installer.

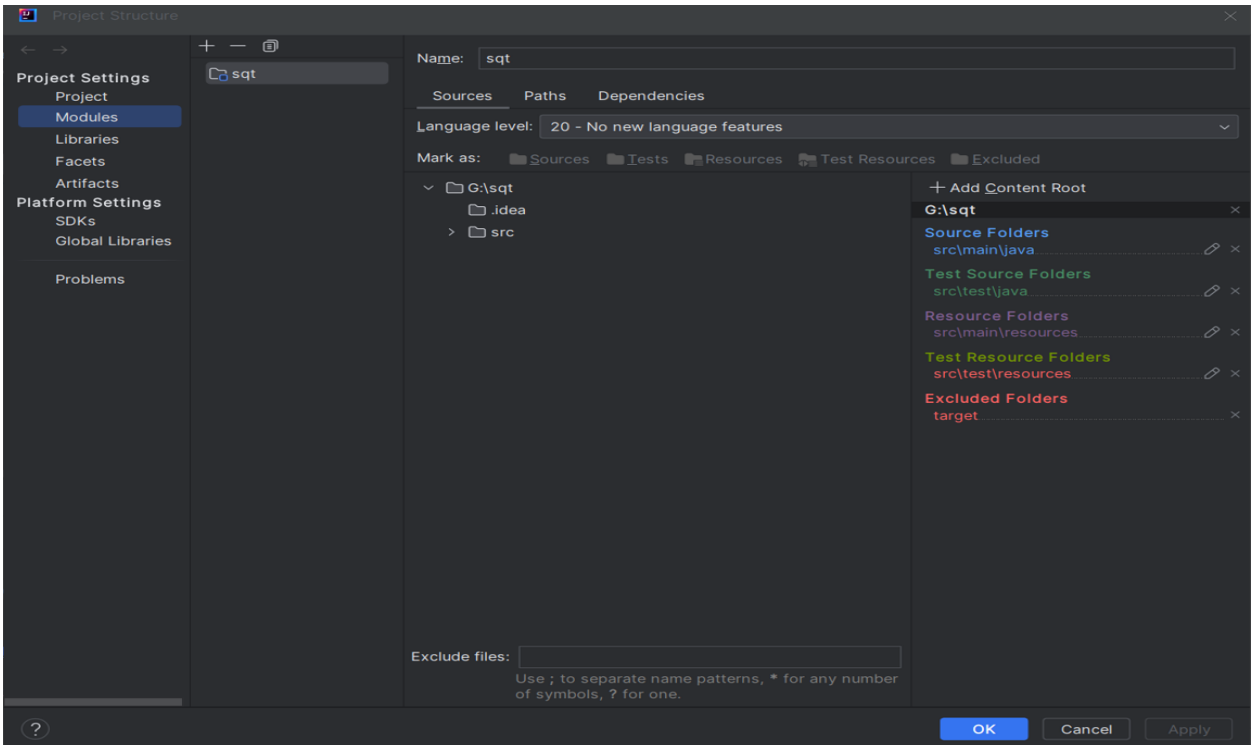

Figure 29: Resources of the project.

Add Maven Dependencies:

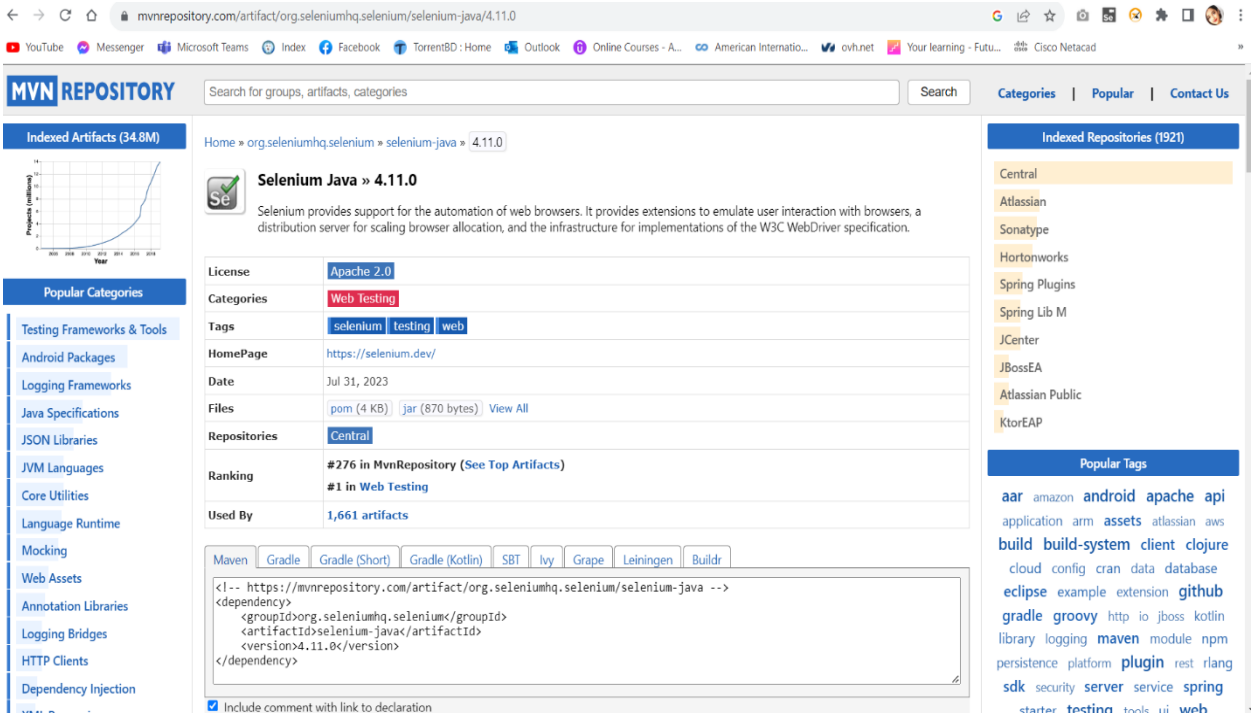

Figure 30: Downloading the Selenium Java-4.11.0.

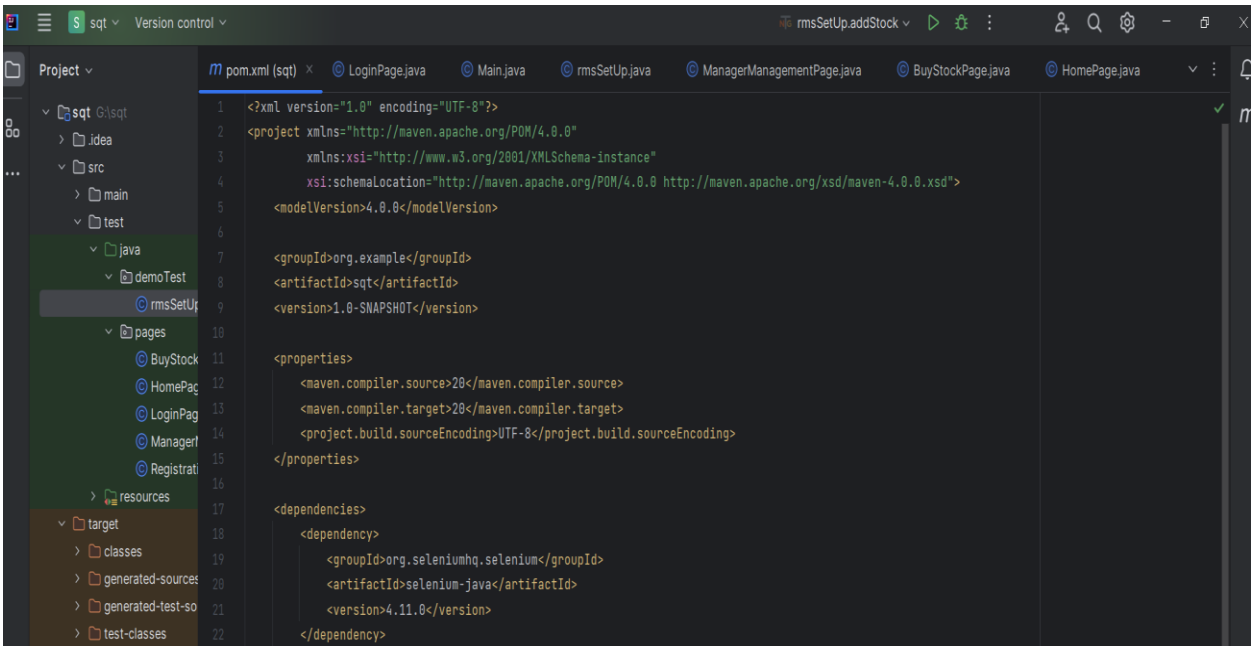

Figure 31: Adding the Selenium Java-4.11.0 dependency in IntelliJ IDEA IDE.

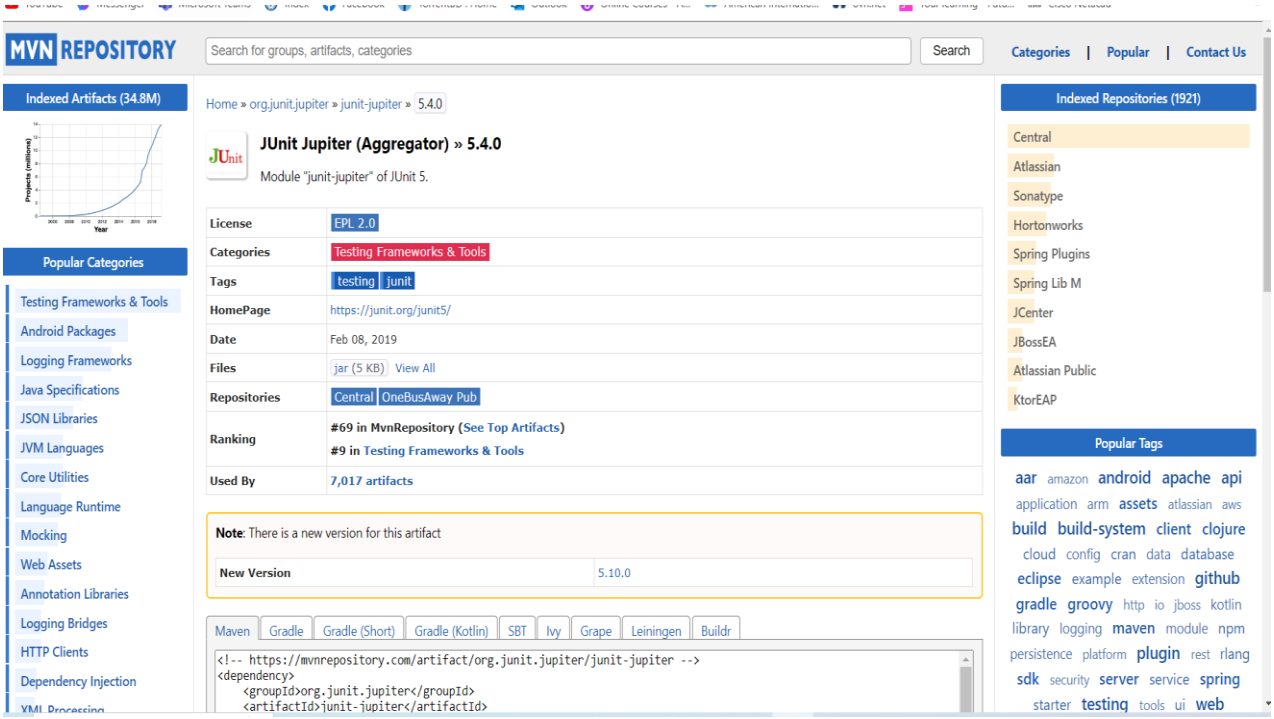

Figure 32: Downloading the Junit Jupiter 5.4.0.

Figure 33: Adding the Junit Jupiter 5.4.0 dependency in IntelliJ IDEA IDE.

Selenium Testing Flowchart:

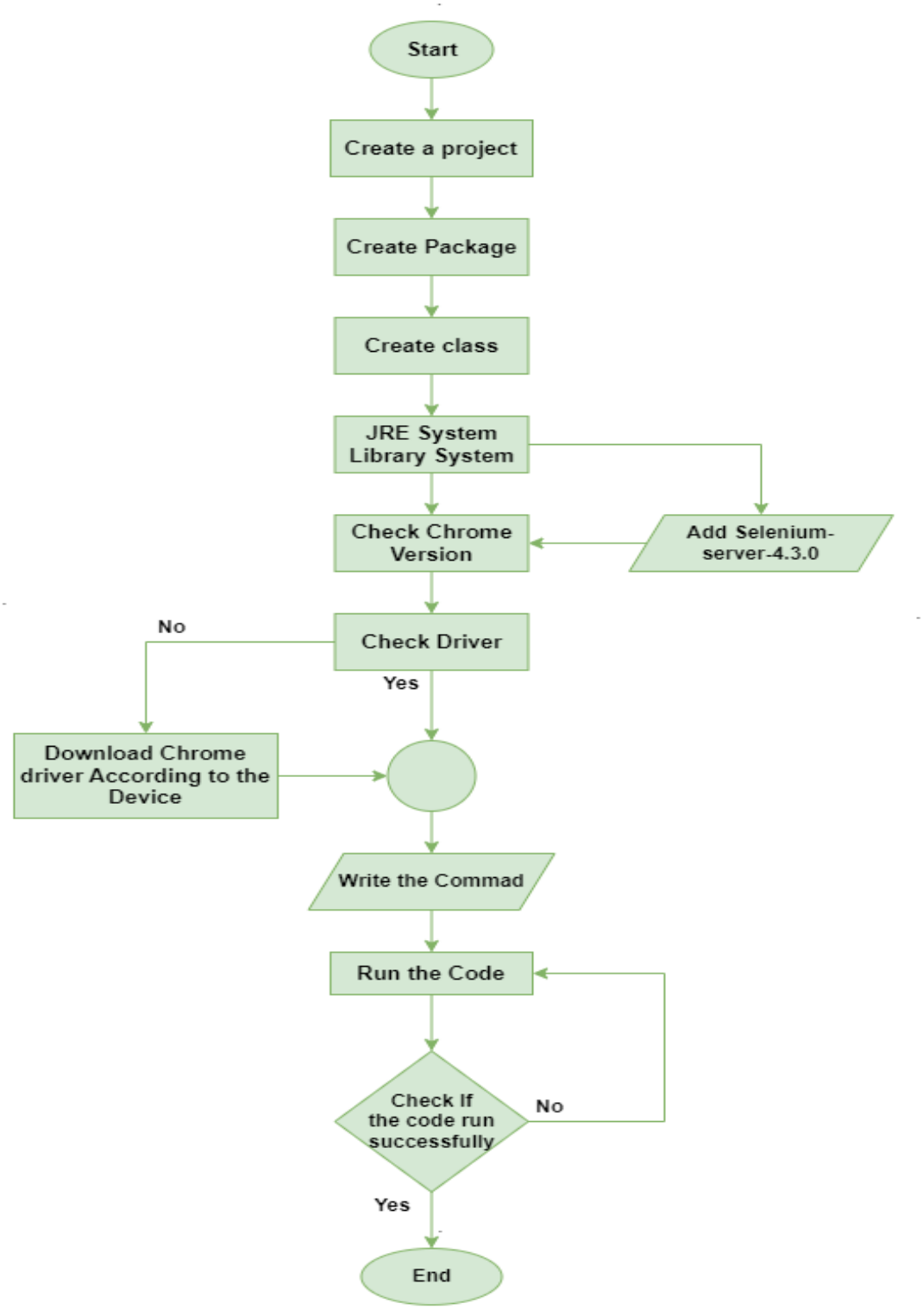

Figure 34: Flowchart of Selenium Testing.

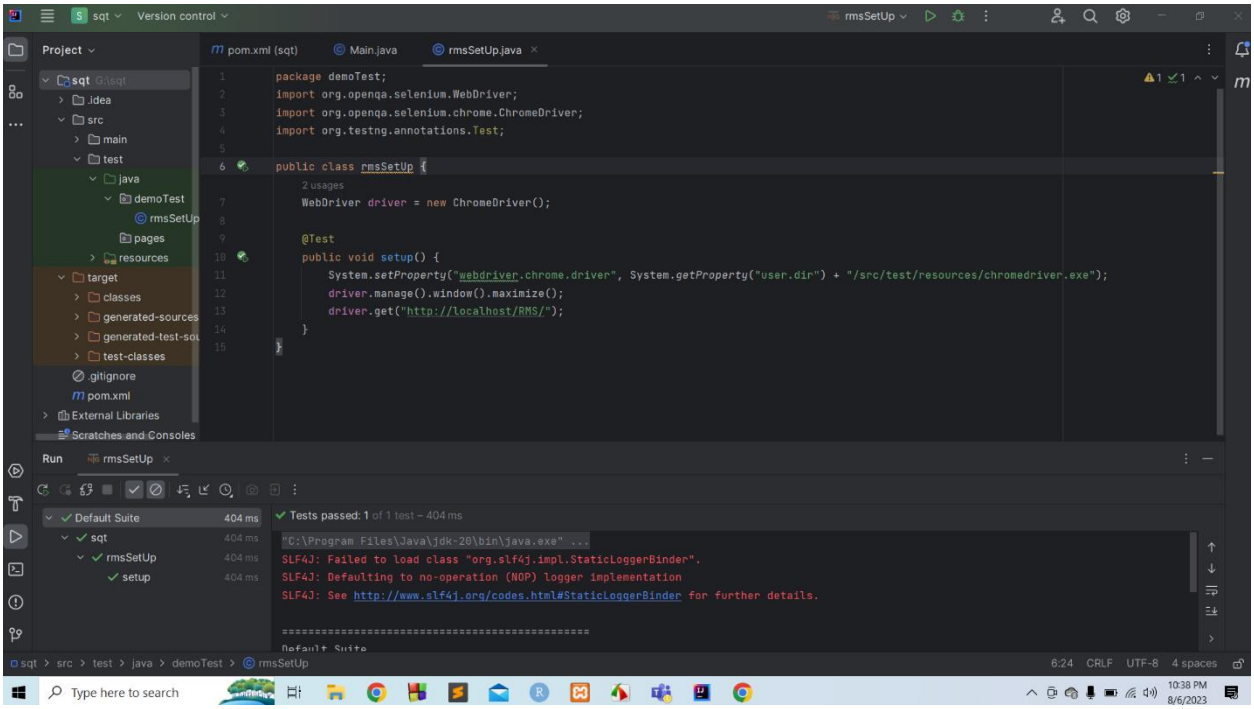

Figure 35: Setup the existing project in the IntelliJ IDEA IDE.

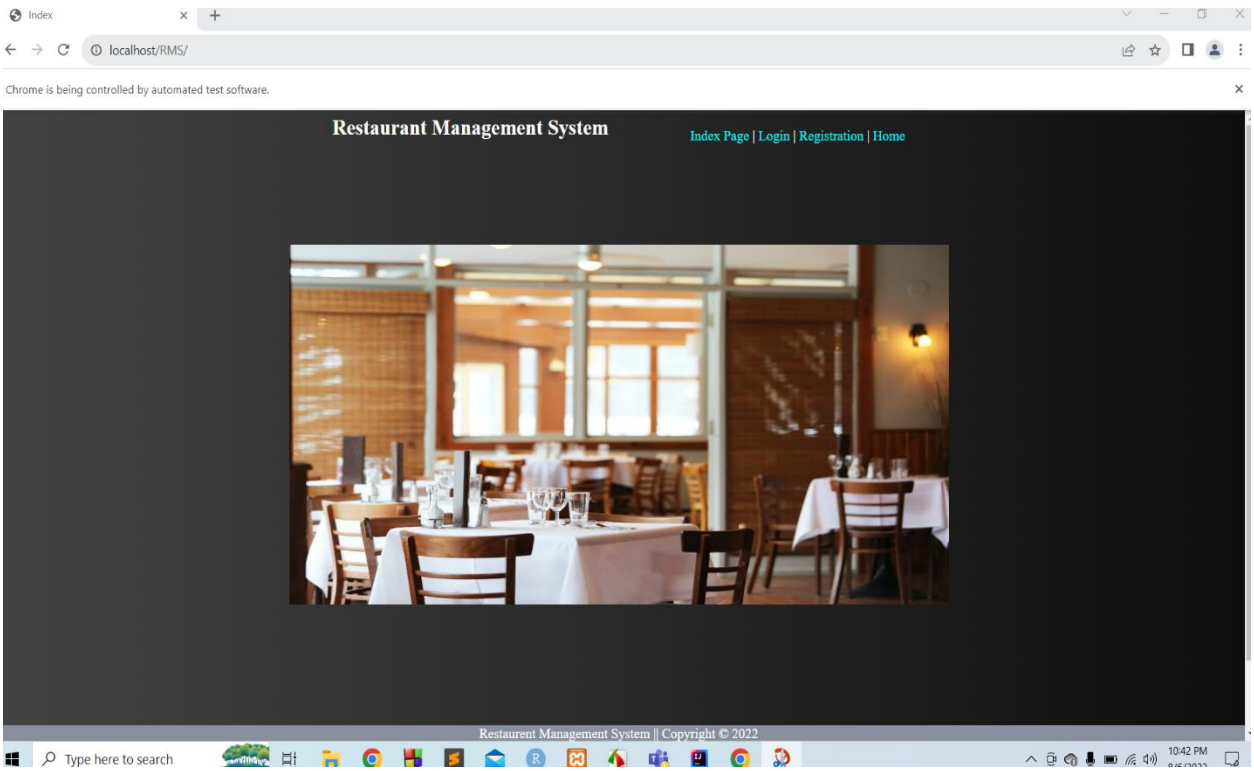

Figure 36: The Restaurant Management System(RMS) existing project ran after automation testing.

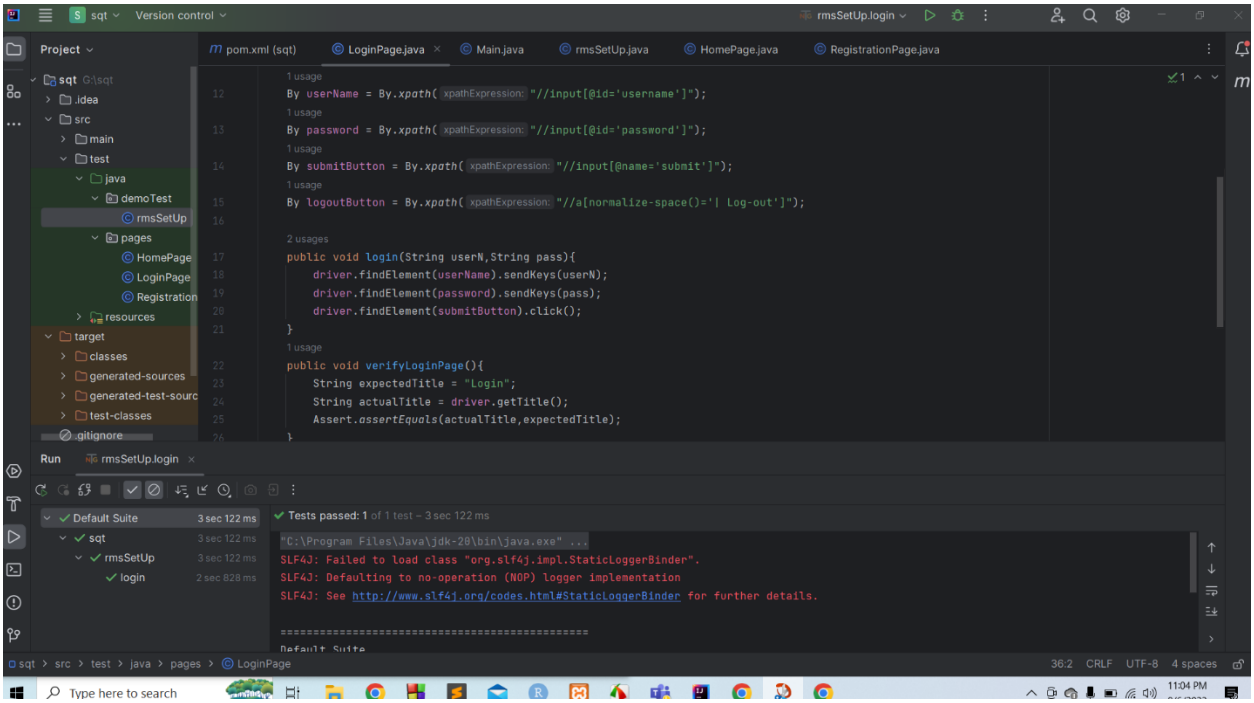

Figure 37: Login page automation test code.

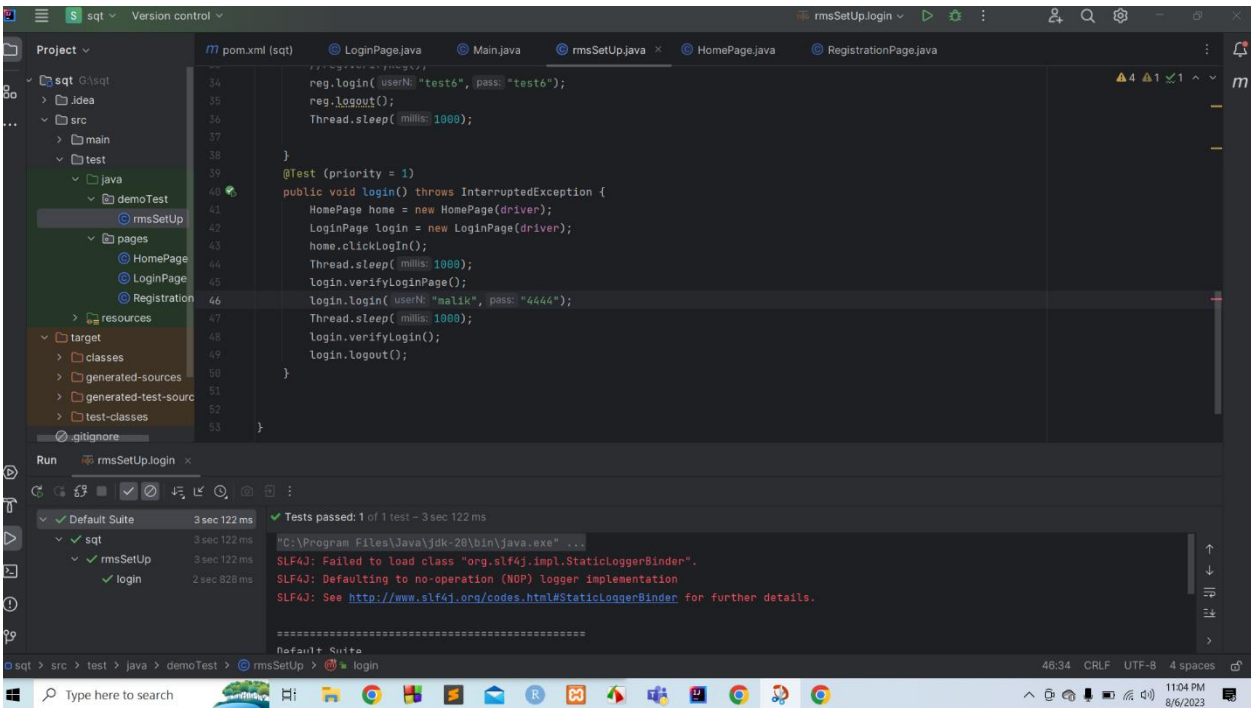

Figure 38: The RMS setup code for Login Page.

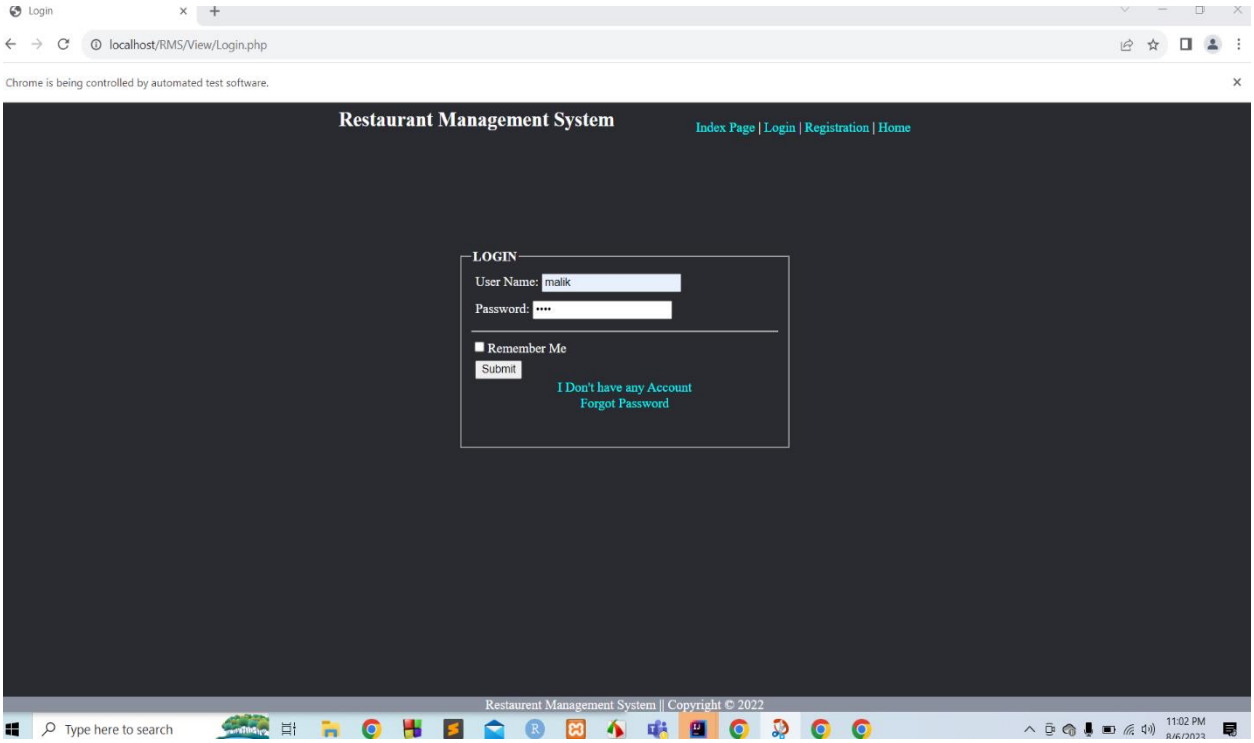

Figure 39: The Restaurant Management System login page ran after automation testing.

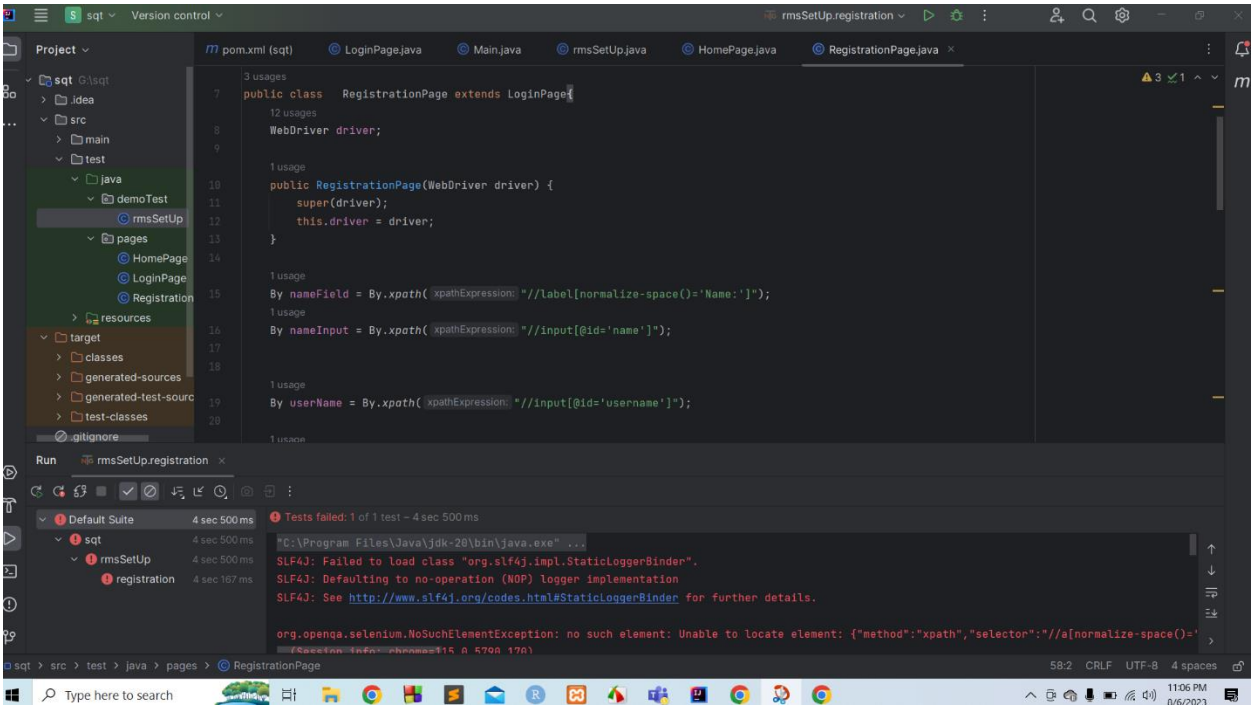

Figure 40: Registration page for automation test code.

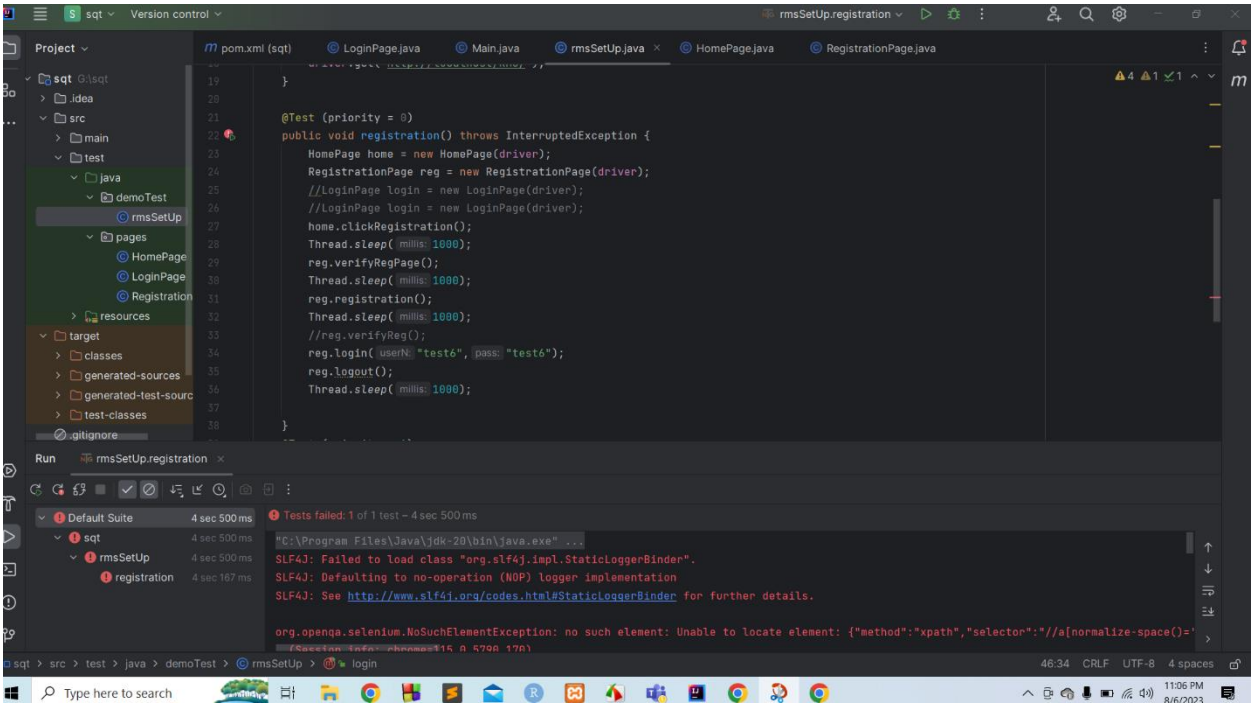

Figure 41: The RMS setup code for Registration Page.

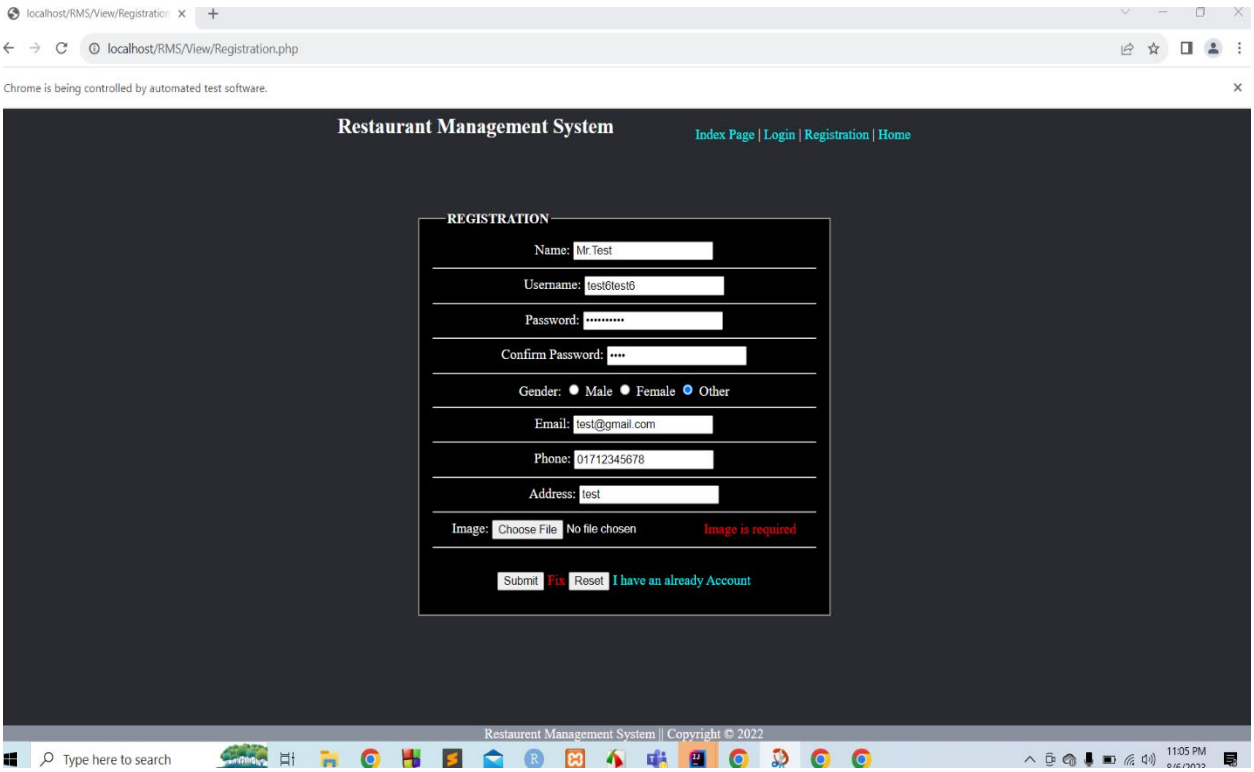

Figure 42: The Restaurant Management System Registration page ran after automation testing.

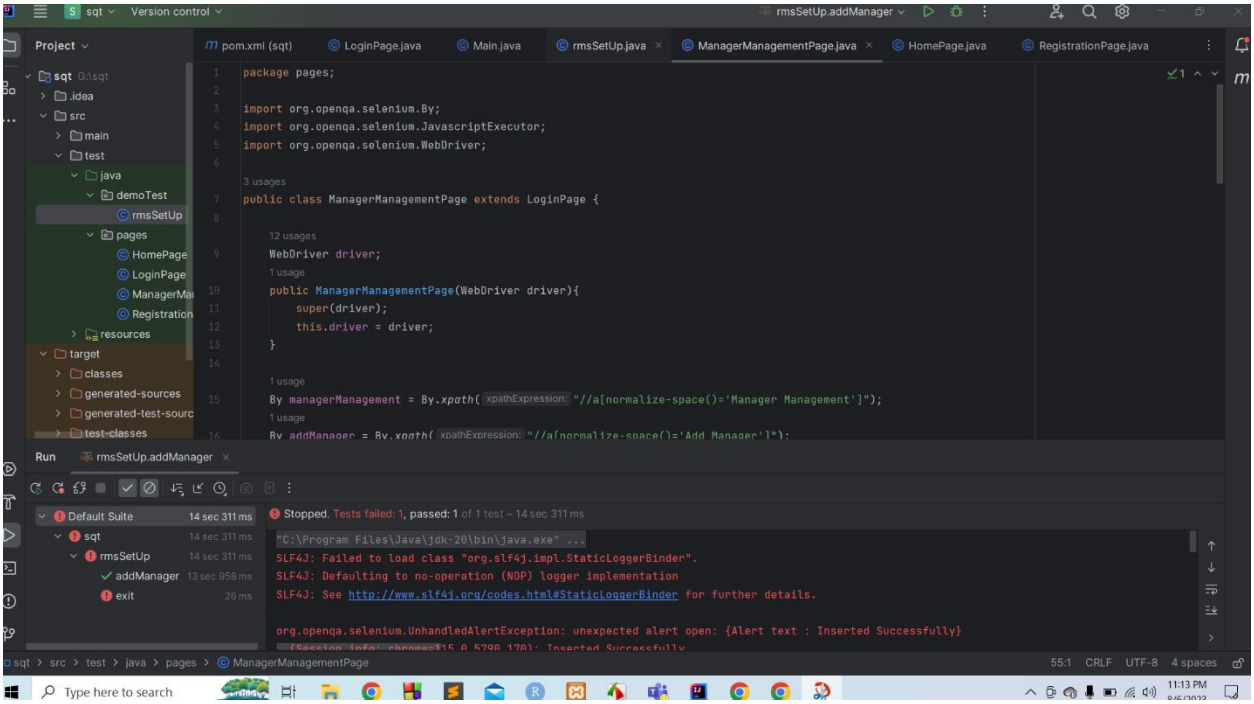

Figure 43: Add Manager page for automation test code.

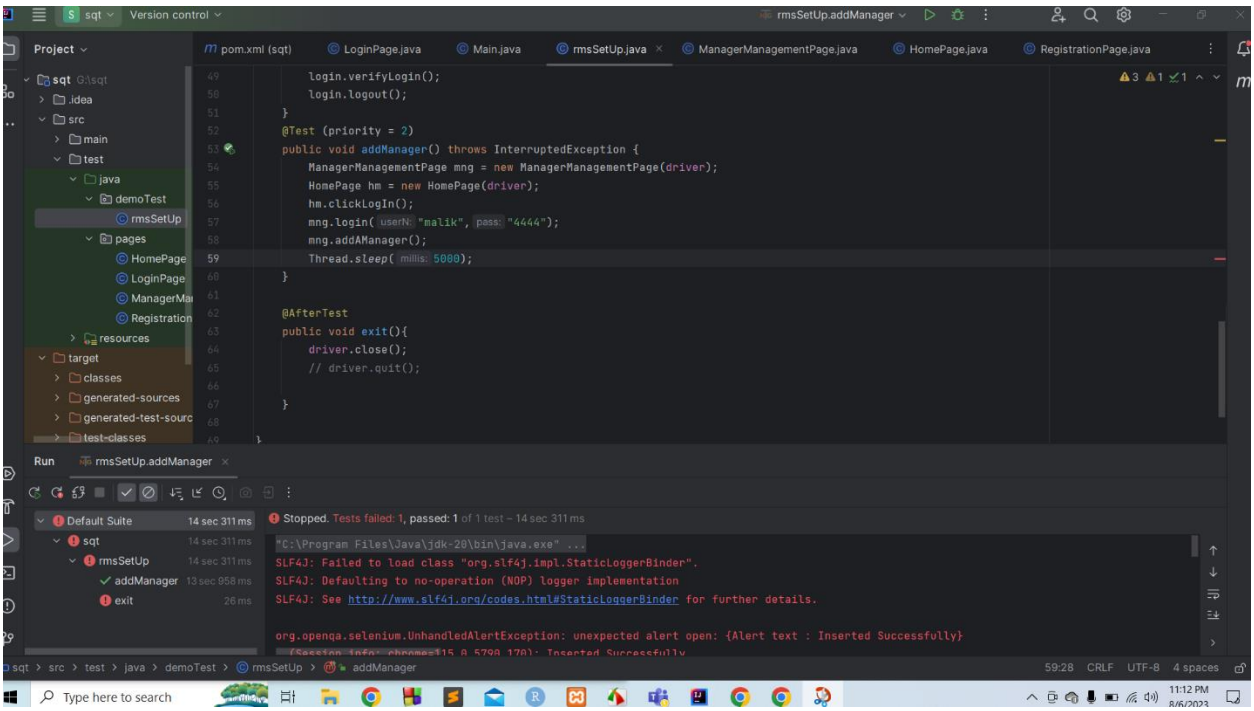

Figure 44: The RMS setup code for Add Manager Page.

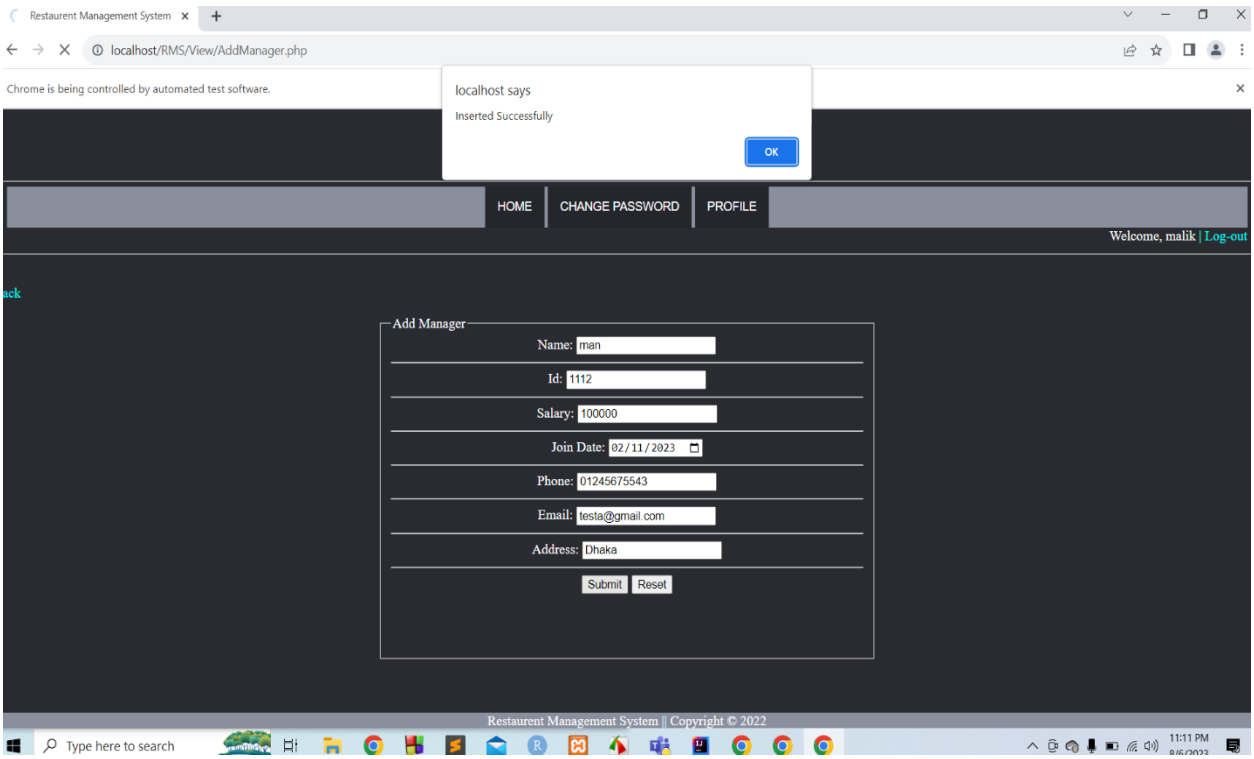

Figure 45: The Restaurant Management System Add Manager page ran after automation testing.

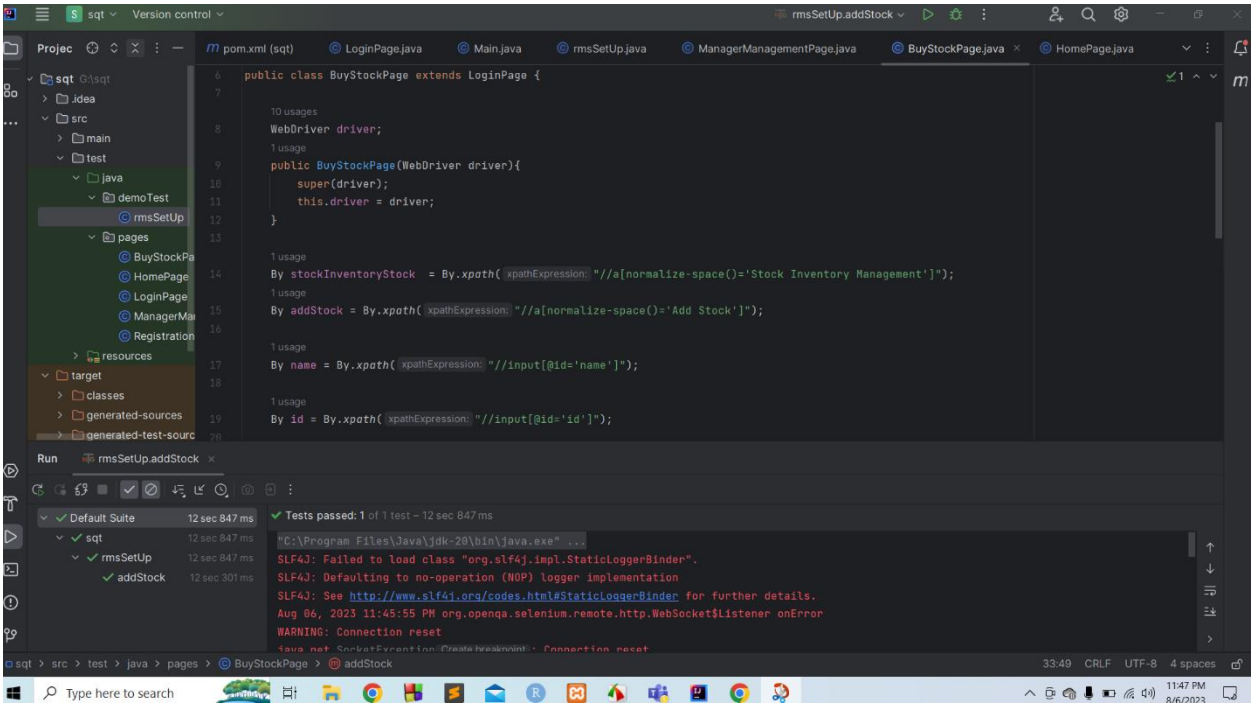

Figure 46: Add Buy Stock page for automation test code.

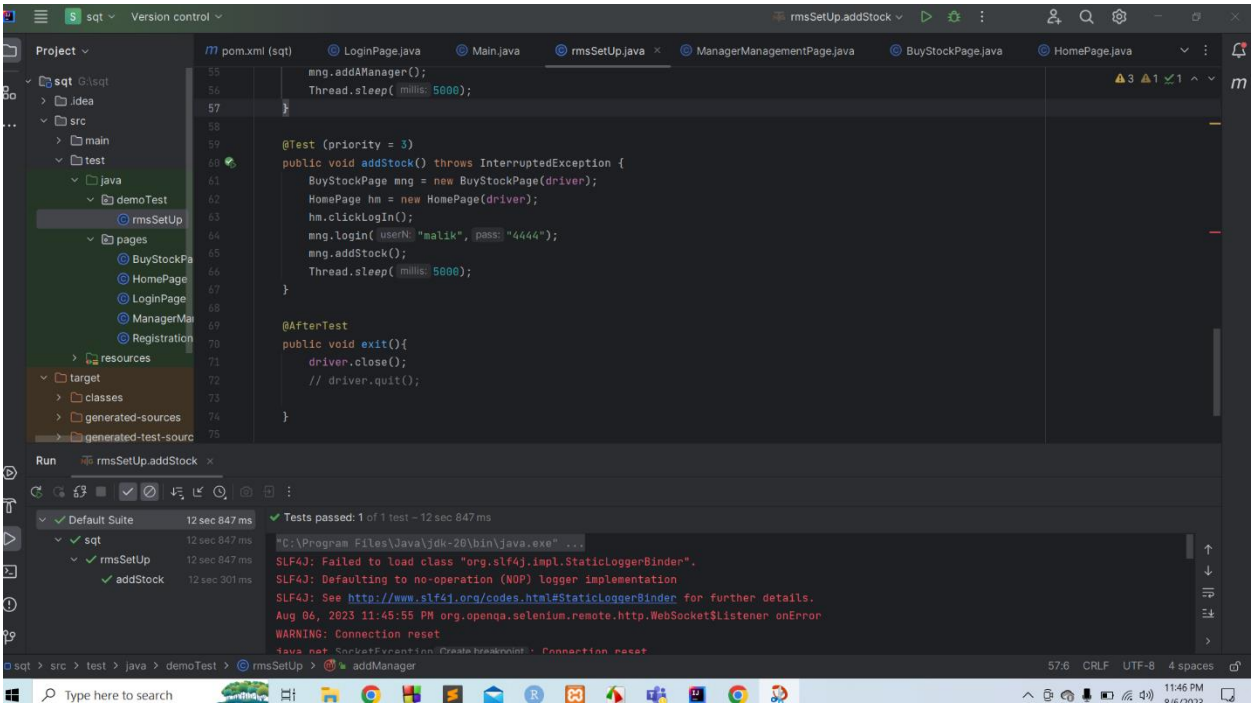

Figure 47: The RMS setup code for Add Buy Stock Page.

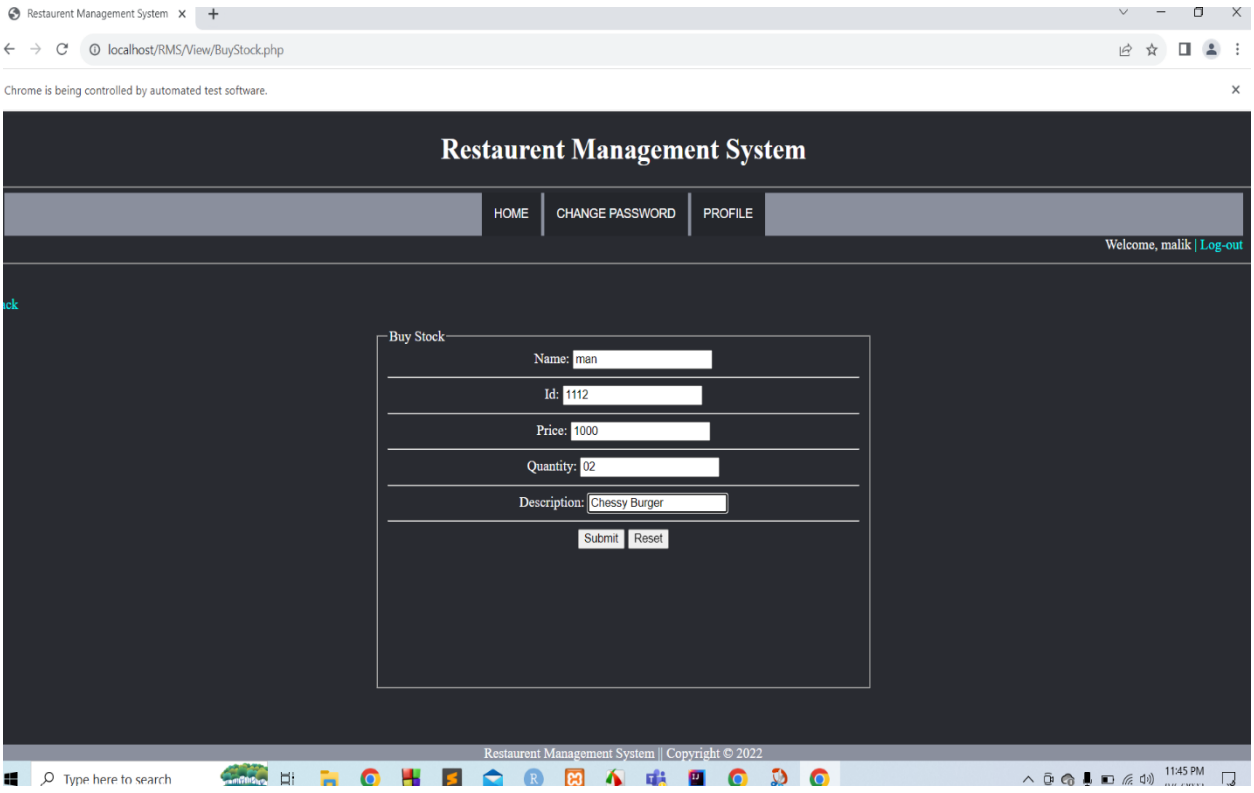

Figure 48: The Restaurant Management System Add Buy Stock page ran after automation testing.

Overview of Selenium Automated Testing with IntelliJ IDEA:

Selenium is a renowned open-source framework that empowers developers to automate web browsers, allowing for efficient and reliable testing of web applications. When paired with the powerful Integrated Development Environment (IDE) IntelliJ IDEA, developers gain a robust platform for creating, executing, and managing automated tests. This integration bridges the gap between software development and quality assurance, fostering better software quality and enhanced user experiences.

Advantages of Using Selenium and IntelliJ IDEA for Automated Testing:

1. **Efficiency and Speed:** The integration of Selenium with the IntelliJ IDEA IDE offers developers a highly efficient way to create and manage automated tests. The IDE's features, such as code completion, intelligent suggestions, and debugging tools, facilitate quicker test script development.

2. **Powerful Automation:** Selenium, being a mature and widely adopted automation framework, provides developers with the ability to automate a wide range of tasks, from simple interactions to complex user flows. This robustness ensures thorough testing of web applications, minimizing the risk of human errors.

3. **Cross-Browser Compatibility:** Selenium supports multiple web browsers, allowing developers to test their applications across various browser environments. This is crucial for ensuring consistent behavior and user experience, especially in a diverse user base.

4. **Integration with Continuous Integration:** Selenium tests created in IntelliJ IDEA can be seamlessly integrated into continuous integration (CI) pipelines, enabling automated testing as a part of the development workflow. This integration ensures that changes are thoroughly tested before they are merged into the main codebase.

5. **Test Reporting and Analysis:** The combination of IntelliJ IDEA and Selenium can be complemented with additional tools, such as TestNG or JUnit, to generate comprehensive

test reports and perform result analysis. This helps in identifying patterns, trends, and potential issues within the application.

Challenges and Considerations:

1. **Initial Setup:** While setting up Selenium in IntelliJ IDEA is relatively straightforward, configuring the necessary dependencies and drivers can sometimes be challenging for beginners. This requires careful attention to ensure smooth execution of tests.

2. **Maintenance Overhead:** Automated tests need periodic maintenance to adapt to changes in the application or browser updates. A change in the structure of web elements or the user interface might require adjustments in the test scripts.

3. **Test Stability:** Automated tests can be sensitive to changes in the web application's layout or structure. Developers need to build robust test scripts that can handle dynamic elements and various scenarios to maintain test stability.

4. **Learning Curve:** For developers new to Selenium or IntelliJ IDEA, there might be a learning curve. However, both have rich documentation and a strong online community, which can help in overcoming these challenges.

Conclusion:

Integrating Selenium automated tests with the IntelliJ IDEA IDE offers developers a potent toolkit for efficient and comprehensive web application testing. The combined features of Selenium's automation capabilities and IntelliJ IDEA's development environment streamline the process of creating, executing, and maintaining tests. This approach ultimately contributes to the delivery of high-quality web applications that meet user expectations, reduce defects, and improve overall software reliability. While challenges exist, they can be overcome with practice, experience, and leveraging the available resources. In a technology landscape where rapid development and high-quality software are paramount, this integration serves as an essential component in the software development lifecycle.