Selenium

with

Java

Table of Content

[What is Software Testing? 2](#_Toc1892155382)

[Manual Testing 2](#_Toc267168910)

[Challenges: Time and Effort Consuming 2](#_Toc1763947954)

[Automation Testing 2](#_Toc1886371269)

[What is Selenium? 2](#_Toc143295691)

[Benefits of Selenium 3](#_Toc2073828500)

[Limitations of Selenium 3](#_Toc1102449479)

[Selenium Versions 3](#_Toc582333263)

[Types of Applications 3](#_Toc593261184)

[How to Run Selenium in OSCAR EMR 4](#_Toc607292615)

[Dependencies Required 4](#_Toc933612081)

[Create the Test 5](#_Toc1578753735)

[Step 1: Create ‘tests’ Folder 5](#_Toc878756100)

[Step 2: Create a Java File for Testing 6](#_Toc729329409)

[Step 3: Generate Test Methods 6](#_Toc1600041423)

[Scenario 1: Login Successfully 6](#_Toc1516374882)

[Scenario 2: Login Failed with Incorrect Username 7](#_Toc1506999993)

[Scenario 3: Login Failed with Incorrect Password 8](#_Toc1631891567)

[Scenario 4: Login Failed with Incorrect Username and Password 9](#_Toc733807106)

[Import Statements 9](#_Toc494207479)

[Insert 500 Error Check 10](#_Toc988845828)

[Switch Current Window 10](#_Toc545491693)

[Extend the Webpage / Window Displays Time 11](#_Toc1907990535)

[Go to Next Record Automatically 11](#_Toc2055713713)

[Select Option from Dropdown Menu 12](#_Toc1884226775)

[Conclusion 12](#_Toc1358204948)

# What is Software Testing?

Software testing is the process of finding defects in a software. The goal is to release quality software to the customer/client.

## Manual Testing

Performing the testing without using any tools is known as Manual Testing. A tester/person will do the testing.

### Challenges: Time and Effort Consuming

* Re-testing: we need to conduct the testing multiple times for the same set of test cases with a different set of data.
* Regression testing: we need to ensure that the existing functionalities are not broken because of the changes done by the developer. We have to conduct regression testing in every cycle (re-testing is part of the regression testing)

## Automation Testing

Performing the testing with the help of tools is known as Automation Testing. The test cases will be transferred to the test script in the automation format.

We need to provide instructions to the tools through the programming language. The language and the internal architecture of the tools can be different, but the way of working is similar. Some tools support multiple languages, like Selenium, which supports Java, Python, JavaScript, etc.

### What is Selenium?

Selenium is a web-based automation tool/library. It supports web-based applications only. It is free, open source, and has a collection of multiple components (IDE – record and playback, WebDriver - automation, Grid - execution). Generally, when people talk about Selenium, that is Selenium WebDriver.

#### Benefits of Selenium

* Open source & free
* Multiple operating systems
* Supports multiple browsers, enables cross-browser testing (Firefox, Safari, Opera, Chrome, Microsoft Edge)
* Support multiple programming languages (Java, Python, C#, Ruby, JavaScript, etc.)
* Integrate third party tools

#### Limitations of Selenium

* Cannot support windows-based applications
* Cannot support excel files, needs to integrate third-party tool Apache POI
* Cannot generate report, needs to integrate third-party tools TestNG and Extent Reports
* Cannot automate graphs and captuas.

#### Selenium Versions

The latest version is Selenium 4.22.0

### Types of Applications

* Web applications: can be accessed through browsers.
* Desktop applications: can download and install on your local systems and use without internet.
* Mobile applications: can download from App store or Play store and install on your mobile devices.

# How to Run Selenium in OSCAR EMR

Since we are running the OSCAR application through VM/Docker containers, we need to download and install the ChromeDriver (127.0.6533.57) manually to the machine to avoid the barrier caused by VM/Docker containers.

For Mac and Linux, install through Homebrew and copy the file path for later.

For Windows, download and install through the URL, then run command ‘xattr -cr 'Google Chrome for Testing.app'’ in the terminal to enable the ChromeDriver.

Or, we can set-up the workflow within the containers, it will require Linux-GUI (example: Ubuntu). It will be more complicated to do now at this point, but we can dig into it more later.

To run the test, ensure it is running in the project local folder, not the container.

## Dependencies Required

All the required dependencies should add in pom.xml

For Selenium Java, the latest version we can use is 3.141.59 since the application is run by Java 8.

*<!--* [*https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java*](https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java) *-->*

<dependency>

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

<artifactId>selenium-java</artifactId>

<version>3.141.59</version>

</dependency>

*<!--* [*https://mvnrepository.com/artifact/org.testng/testng*](https://mvnrepository.com/artifact/org.testng/testng) *-->*

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>7.3.0</version>

<scope>test</scope>

</dependency>

*<!--* [*https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager*](https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager) *-->*

<dependency>

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

<artifactId>webdrivermanager</artifactId>

<version>3.8.1</version>

</dependency>

*<!--* [*https://mvnrepository.com/artifact/com.google.guava/guava*](https://mvnrepository.com/artifact/com.google.guava/guava) *-->*

<dependency>

<groupId>com.google.guava</groupId>

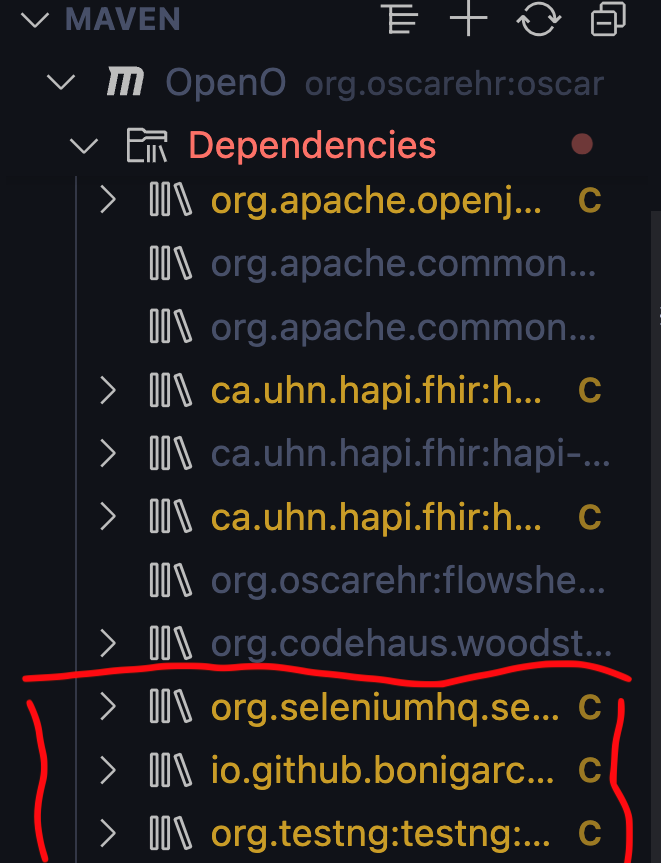
<artifactId>guava</artifactId>

<version>25.1-jre</version>

</dependency>

## Create the Test

After adding the required dependencies and installed the Chrome Driver, ensure the dependencies are added successfully.



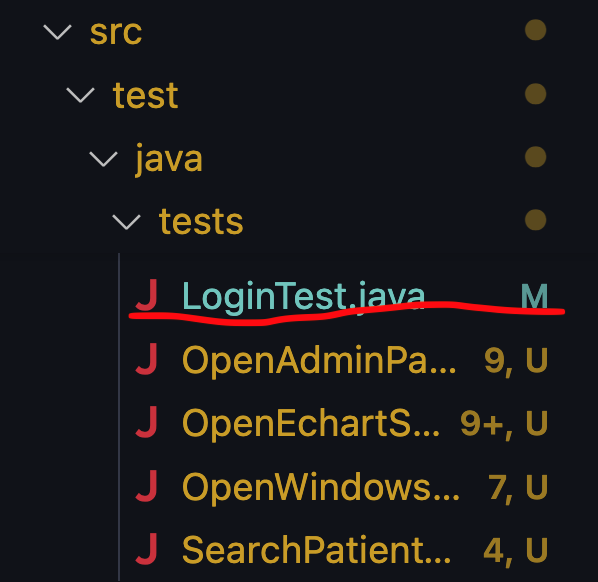
### Step 1: Create ‘tests’ Folder

Create a folder named ‘tests’ in src -> test -> java directory.



### Step 2: Create a Java File for Testing

Create a Java file for testing in the tests directory. For example, if you want to test the login page, you can create a Java file named ‘LoginTest.java’.



### Step 3: Generate Test Methods

After creating the LoginTest.java file, you need to consider how many test cases (test methods) you are going to create.

For login page, there should be at least 4 scenarios – login successfully, login failed with incorrect username, login failed with incorrect password, and login failed with incorrect username and incorrect password.

#### Scenario 1: Login Successfully

*Note:* ensure you set the property of Chrome Driver using the local path.

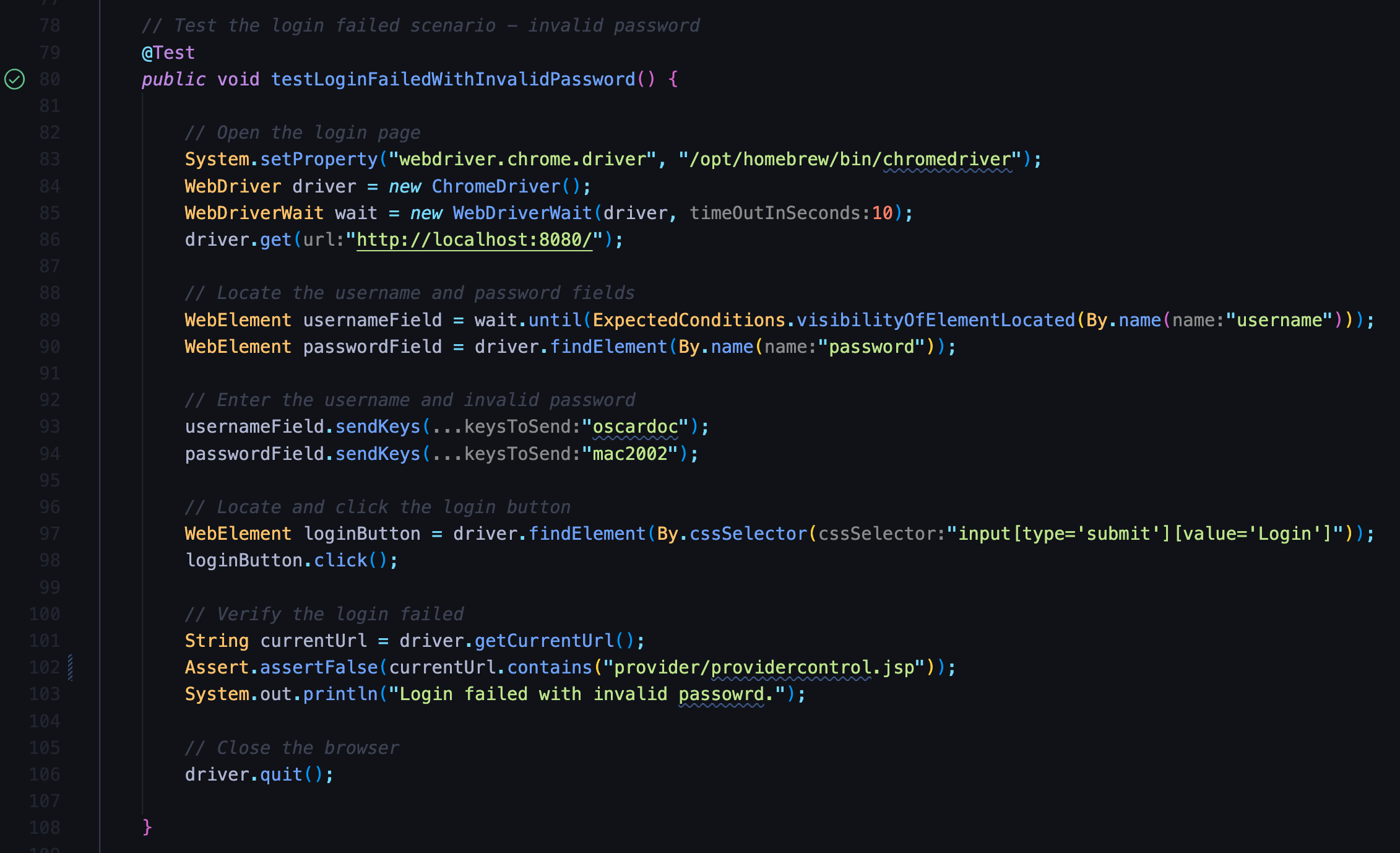
*Keypoints:*

* You need to set the login page URL so the browser knows where to start (driver.get(“http://localhost:8080/”)).
* You need to set the correct name/id of the username and password so the browser knows which field needs to be filled.
* You need to set the correct username and password, let the browser enter the information to the indicated fields.
* You need to set the login button correctly (in the example above, I use cssSelector to find the login button; this is not the only approach).
* You need to set the expectations (e.g. expect URL) to ensure that the test passes / fails (Assert.assertTrue(...)).

#### Scenario 2: Login Failed with Incorrect Username

*Keypoint:* you need to set the incorrect username intentionally, let the browser enter the incorrect username.

#### Scenario 3: Login Failed with Incorrect Password

*Keypoint:* you need to set the incorrect password intentionally, let the browser enter the incorrect password.

#### Scenario 4: Login Failed with Incorrect Username and Password

*Keypoint:* you need to set the incorrect username and password intentionally, let the browser enter the incorrect username and password.

#### Import Statements

Ensure all the required import statements are added in the header (some import statements will add automatically when the code generates).

## 

## Insert 500 Error Check

Since we randomly encounter a 500 error when opening the webpage / window, it is crucial to check the 500 error after the webpage / window opens.

*Keypoint:* use try-catch to check if any keyword related to the 500 error appears on the webpage / window. If there is one, display the 500 error message in the terminal, and the test fails. If not, pass the 500 error check and pass the test.

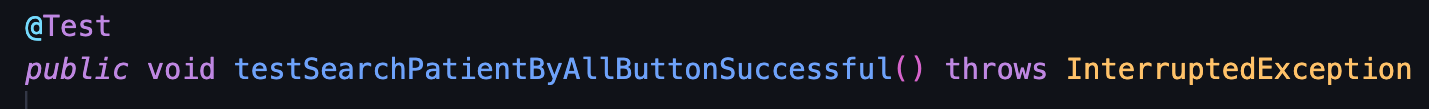
## Switch Current Window

Most of the time, when we open a webpage by clicking the tab / button, a new window appears. If you encounter this situation, you need to write code to switch the new-opened window to the current window. This is a crucial step in case you want the testing to continue working on the new window or you want to set any assert condition to check if the window is opened correctly.

*Keypoint:* to ensure that the window has been switched, you can insert a line of code to check the current URL (System.out.println("The current URL is: " + currentUrl);). If the URL matches the new window, you can go ahead.

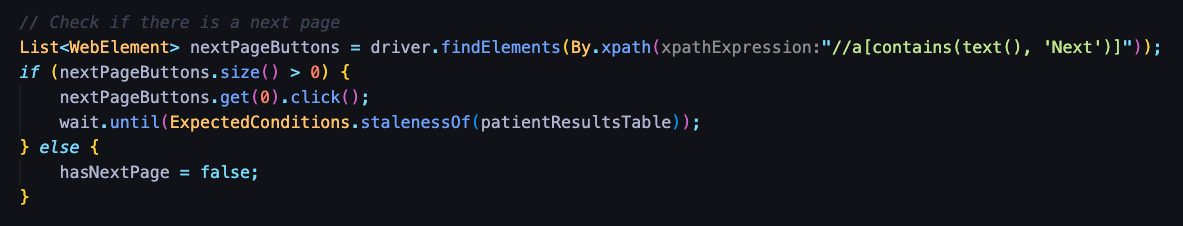
## Extend the Webpage / Window Displays Time

To ensure the webpage / window opens with the desire output, it is helpful to let the webpage / window stays there for a few seconds before quitting the browser.

*Keypoint:* the ’throws InterruptedException’ enables the ’Thread.sleep’ method.

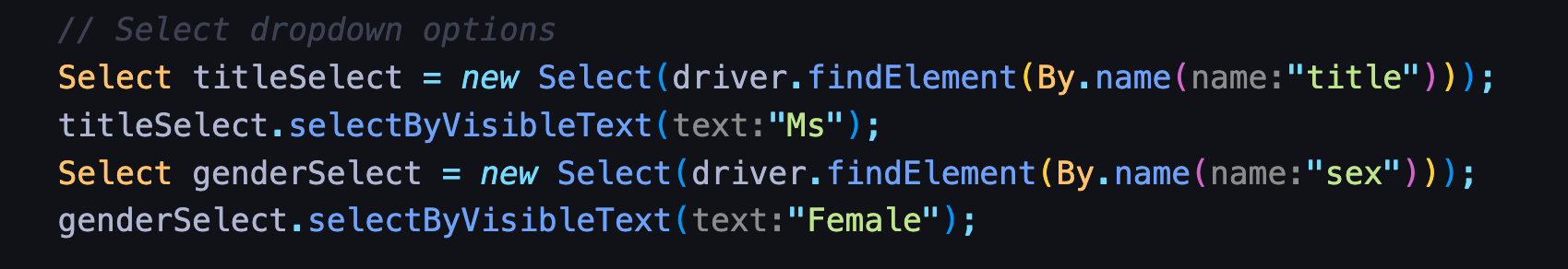
## Go to Next Record Automatically

This is the crucial action when testing opening the patients’ eCharts and their master records. By doing this, the browser can open each eChart and master record following the table’s order automatically (SearchPatientTest.java - ‘testOpenAllEchartsSuccessful()’ and ‘testOpenAllMasterRecordsSuccessful()’).

*Keypoint:* use the while loop and for loop to ensure the browser can look for patients following the table’s order. Use the if statement to check for any ’Next’ button to go to the next page after opening all patients’ charts / master records on the current page.

## Select Option from Dropdown Menu

This scenario can happen when filling out the form, such as testing creating demographic records. Sometimes, you must enable the browser to select an option from the dropdown menu.

*Keypoint:* use ’Select’ method, ensure you provide the correct name / id of the dropdown menu, then the browser will be able to select the visible option you indicate in the method.

## Conclusion

The purpose of running selenium testing is to execute simple tests like opening the webpage / window without encountering a 500 error, creating a demographic record successfully, etc. Running the automated UI testing saves developers time and reduces testing errors compared to manual testing.