Table of Contents

[1 Document General 3](#_Toc133229950)

[1.1 Glossary 3](#_Toc133229951)

[2 Installation Steps 4](#_Toc133229952)

[2.1 Programs to be downloaded and installed. 4](#_Toc133229953)

[2.1.1 Download and install Java JDK 19 4](#_Toc133229954)

[2.1.2 Download and install Java JRE 5](#_Toc133229955)

[2.1.3 Download and install Eclipse 5](#_Toc133229956)

[2.1.4 Download and install Android Studio 5](#_Toc133229957)

[2.1.5 Download and install Appium Server Gui 7](#_Toc133229958)

[2.1.6 Download and install Appium Inspector 7](#_Toc133229959)

[3 Naming Conventions 8](#_Toc133229960)

[3.1 Java project folder structure: 8](#_Toc133229961)

[3.2 E2e Spec Files 8](#_Toc133229962)

[3.3 Elements 8](#_Toc133229963)

[3.4 Page Objects 9](#_Toc133229964)

[3.5 Variables 10](#_Toc133229965)

[3.5.1 Transaction Variables 10](#_Toc133229966)

[3.5.2 Personal Variable file 10](#_Toc133229967)

[3.5.3 Variables.Utils: 11](#_Toc133229968)

[4 Support Package Explanations 12](#_Toc133229969)

[4.1 Commands 12](#_Toc133229970)

[4.1.1 Click Element 12](#_Toc133229971)

[4.1.2 enterFieldValue 12](#_Toc133229972)

[4.1.3 clickDropDownValue 12](#_Toc133229973)

[4.2 databaseHelper 12](#_Toc133229974)

[4.2.1 retrieveDBData 12](#_Toc133229975)

[4.2.2 closeConnection 12](#_Toc133229976)

[4.3 driverHelper 13](#_Toc133229977)

[4.3.1 getDriver 13](#_Toc133229978)

[4.3.2 openWebsite 13](#_Toc133229979)

[4.3.3 closeDriver 13](#_Toc133229980)

[4.4 reportManager 13](#_Toc133229981)

[4.4.1 reportSetup 13](#_Toc133229982)

[4.4.2 createReport 13](#_Toc133229983)

[4.5 waitHelper 13](#_Toc133229984)

[5 Selenium Grid 4 14](#_Toc133229985)

[5.1 Download Jar file: 14](#_Toc133229986)

[5.2 Create local folder: 14](#_Toc133229987)

[5.3 Download browser drivers: 14](#_Toc133229988)

[5.4 Grid Startup: 15](#_Toc133229989)

# Document General

## Glossary

The table below contains a list of definitions, acronyms and abbreviations used in this document:

|  |  |
| --- | --- |
| Acronym / Abbreviation | Definition / Description |
| SVN | Subversion (version control system)/ Software Version Number |
| Java |  |
| JDK | Java Development Kit |
| JRE | Java Runtime Environment |
| Eclipse IDE | Java-based development platform where the code is created. |
| TestNG | TestNG is designed in such a way that it covers all the categories of tests comprising unit, functional and integration. |
| ExtentReports | A plugin that creates reports for tests that run |
| Appium Server | Creates a connection between automation Test and emulator or mobile device |
| Appium Inspector | Inspector is used to locate the elements on emulators or mobile devices |
| Android Studio | Is used to set up an emulator. |

# Installation Steps

## Programs to be downloaded and installed.

### Download and install Java JDK 19

Download the JDK from: <https://www.oracle.com/java/technologies/javase/jdk19-archive-downloads.html>:

Install the JDK and then set the system path:

Open Environment Variables on your computer:

1. Create a system Variable called JAVA\_HOME with the value of where the jdk is located.

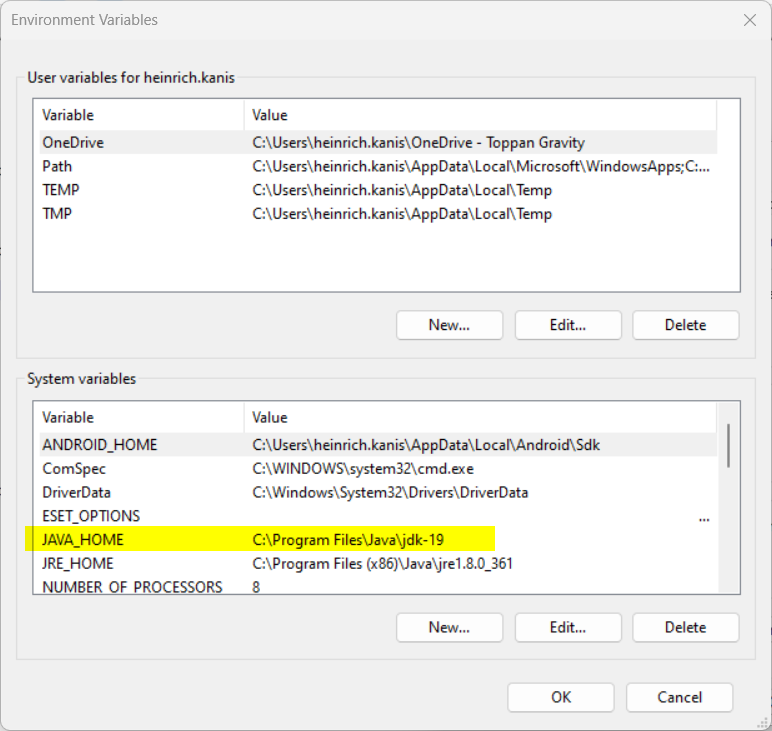
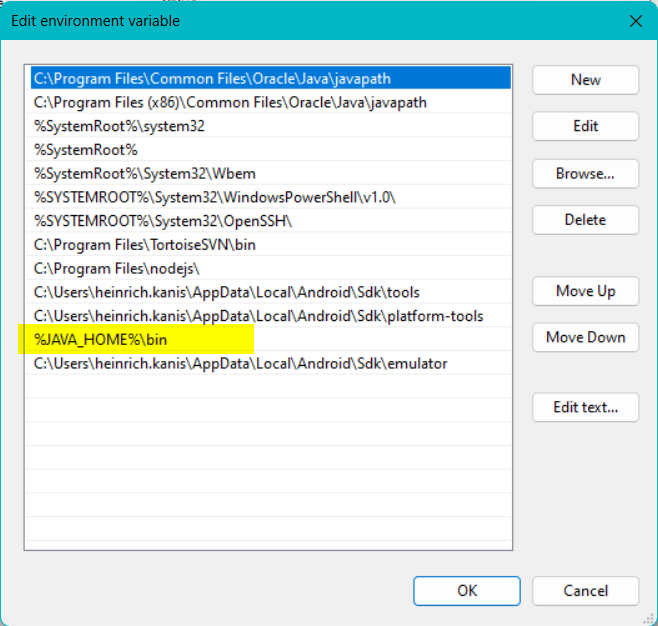


Figure 1 Java Home System Variable

1. Edit the Path and the JAVA\_HOME with the Value.



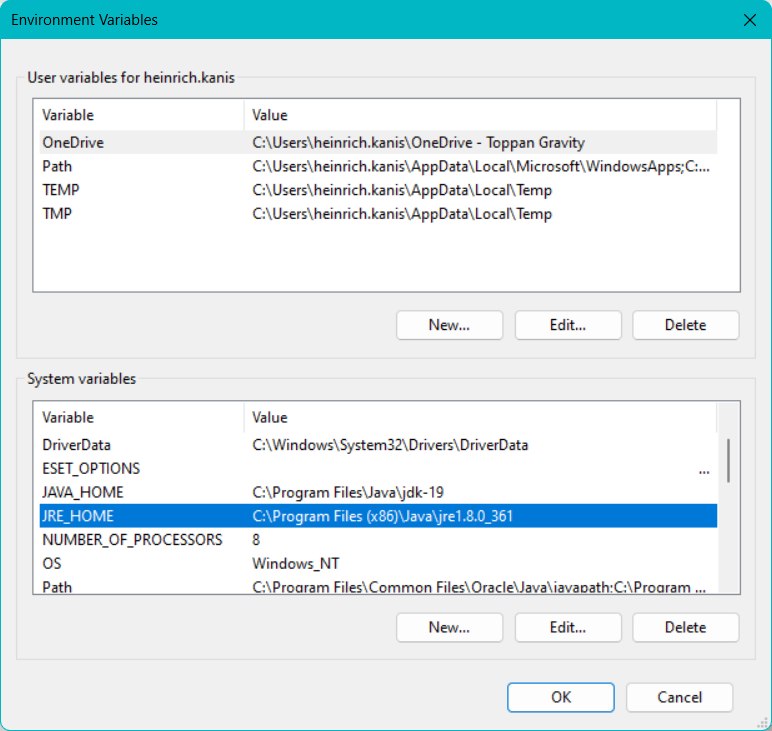
### Download and install Java JRE

Download the JRE 1.8 online from: <https://www.java.com/en/download/manual.jsp>

Install the JRE and then set the system path:

Open Environment Variables on your computer:

1. Create a system Variable called JRE\_HOME with the value of where the jre is located.



### Download and install Eclipse

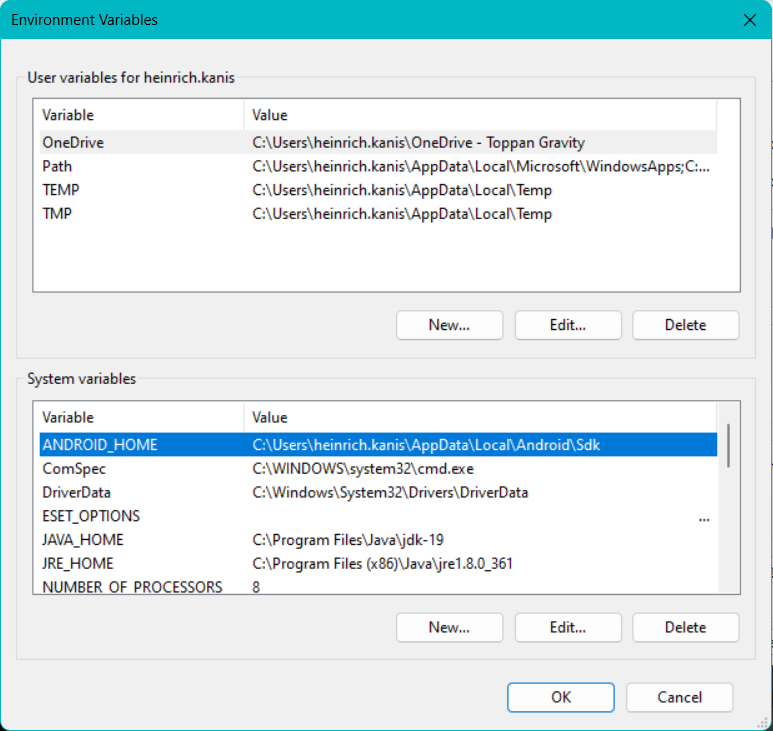
Download and install eclipse from https://eclipseide.org/

### Download and install Android Studio

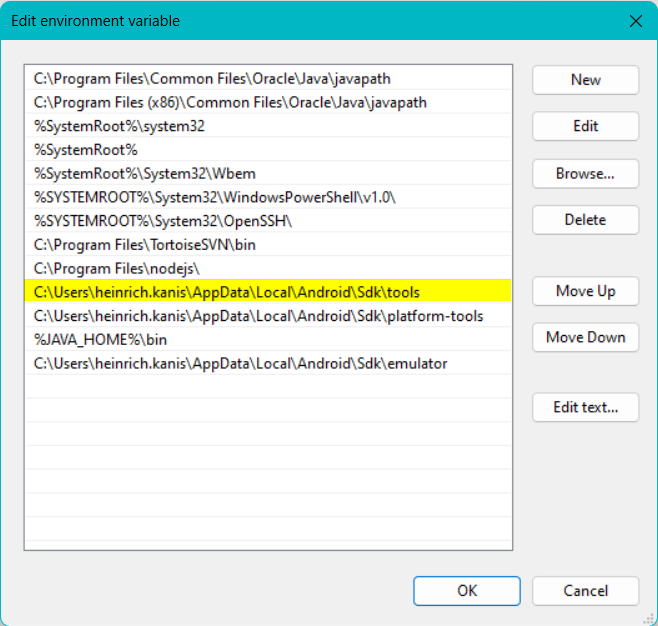
Download and install Android Studio from <https://developer.android.com/studio>

Open Environment Variables on your computer:

1. Create a system Variable called ANDROID\_HOME with the value of where the Android Home is located.



1. Edit the Path and the path to the Android Tools.



1. Edit the Path and the path to the Android Platform Tools.

Graphical user interface, text, application, email

Description automatically generated

### Download and install Appium Server Gui

Download and install Appium Server from <https://www.automationtestinghub.com/appium-desktop/>

Address = 127.0.0.10

Port = 1097

BasePath = /wd/hub

PS these configurations need to be used so that the selenium core can connect to the Appium server.

### Download and install Appium Inspector

Download and install Appium Inspector from <https://github.com/appium/appium-inspector/releases>

The following setting need to be added for the inspector to connect to the Appium server:

Remote host = 127.0.0.10

Remote port = 1097

Remote path = /wd/hub/

The following desired capabilities needs to be set according to the emulator being used:

"platformName": "Android",

"appium:app": "",

"appium:deviceName": "pixel4-api",

"appium:automationName": "UIAutomator2",

"appium:platformVersion": "12"

# Naming Conventions

## Java project folder structure:

The basic folder structure: \***Please note that this does not include the location for the “C** **ORE” files which are also referenced using relative paths**\*

\ {Project Name}

\ {src/main/java}

\ e2e [This is where all the automation test are created]

\ elements [This is where the element files are located for each TC’s]

\ fixtures [This where attachments are saved for test cases]

\ page objects [This is where all the components are created for the TC’s]

\ support [This is where all the selenium methods are created and stored]

\ variables [This is where personal variable files are saved and TC variables]

\{extentReports} [This is where the report configuration file is located]

## E2e Spec Files

A folder will exist Named Transaction, under this folder a sub folder will be created for the transaction being done. The spec file will be added into the sub folder.

The spec files (Automation Test Case) will be named as follow:

Senario1.java

**Use the example.java file in the e2e folder as an example.**

## Elements

An element file will be created per component. This is to ensure reusability across all tests for the project. For example: a file will be created for the Person Detail where all elements on this component will be stored, the person creating a test case then does not need to create another elements file for the person detail.

The file name of the file will be the Component Name.

A class needs to be created in the file, and this class name will be the same as the File name.

The element names in this file will be created to the following format:

* Text boxes and input fields à txt + Fieldname [ txtUserName ]
* Buttons à btn + Button name [ btnContinue ]
* Dropdowns à drp + Dropdown name [ drpCountry ]
* Checkboxes à chk + Checkbox name [ chkPresent ]
* Labels à lbl + Label Name [ lblNew ]
* Status à sts + Status Name [ stsPresent ]
* Date à Picker dtp + Date Picker Name [ dtpExpiryDate ]

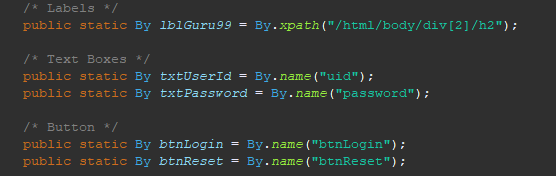


Figure 2 example.java

**Use the example.java file in the elements folder as an example.**

## Page Objects

A page object file will be created per screen. This page object file will then be called from your spec file to setup the transaction.

* File Name & Class name à The name of the screen login\_pom.java

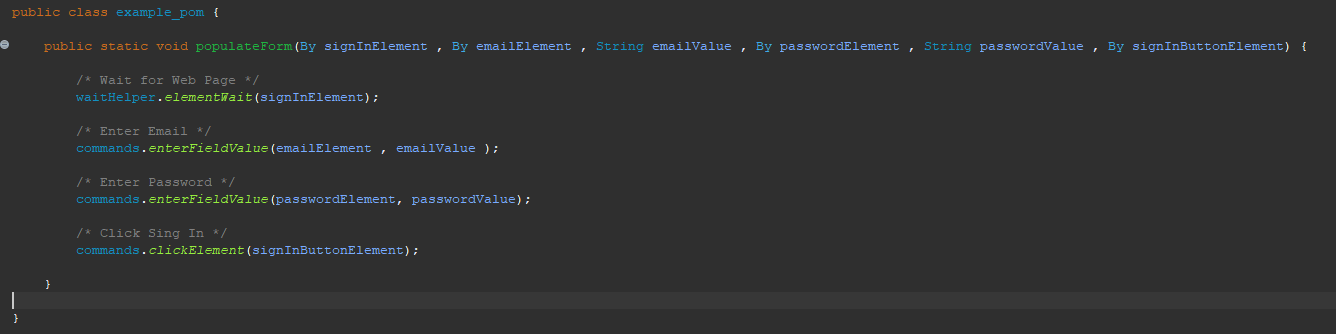


Figure 3 example\_pom.java

**Use the example\_pom.java file in the page objects folder as an example.**

## Variables

There are 3 different variable files that will be used for the automated testing.

### Transaction Variables

A variable file will be created per Spec. The folder structure will be the same as that of the e2e Spec files.

Eg.

Variables

Transaction [Folder]

Login [Folder for Type of Transaction]

Scenario1\_var.java [This indicates that this is only for Scenario 1]

Text

Description automatically generated

Figure 4 transaction variable file

**Use the example\_var.java file in the variables.transaction folder as an example.**

### Personal Variable file

A separate variables file will must be created for each person, this will contain all the specific variable for that individual. The file will be named as variables\_Initials, the owner of this file will add their initials.

The class name will be the same as the file name.

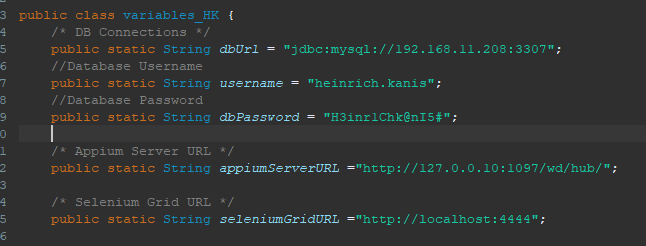


Figure 5 variables\_HK.java

**Use the variables\_HK.java file in the variables folder as an example.**

### Variables.Utils:

This is a utility file that stores all the desired capabilities for Appium and Appium Server. This file will be used when doing automated testing on an emulator or mobile device.

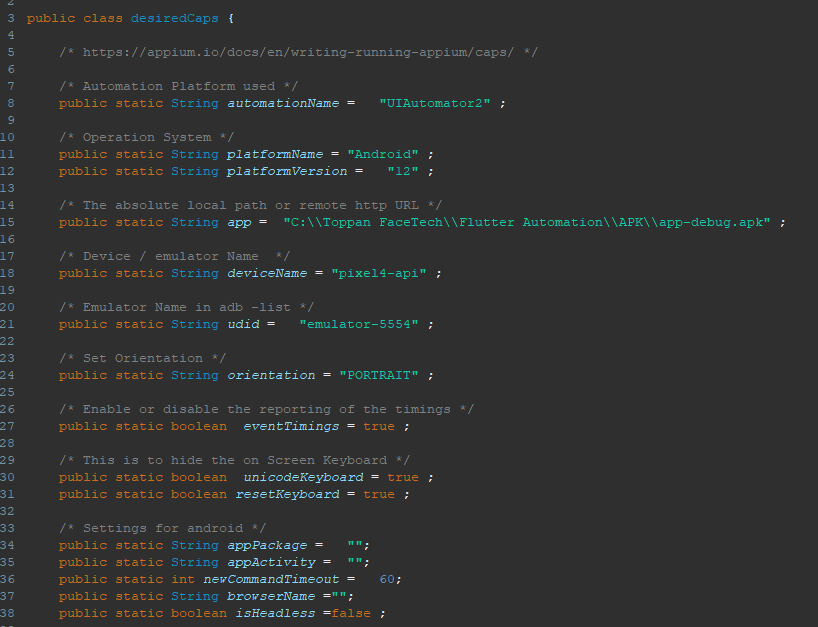


Figure 6 variables.utils

**Use the desiredCaps.java file in the variables.utils folder as an example.**

# Support Package Explanations

In the Support package there are different helpers, each helper will no be discussed individually.

## Commands

The commands file has different classes in the file and each command is create for a specific purpose. New commands may be added if there is a need.

### Click Element

This command is used to click any element / button on the screen.

To use the command simply use the following: commands.clickElement();

### enterFieldValue

This command is used to enter a value into the field.

To use the command simply use the following: commands.enterFieldValue();

### clickDropDownValue

This command is used to select a value from a dropdown.

To use the command simply use the following: commands.clickDropDownValue();

## databaseHelper

This class is used to establish a connection to the Database which can then return a value from the DB using a select statement which can then be used in the automated test.

### retrieveDBData

This class opens the connection to the DB and can return anything from the DB if the query supplied is executed correctly.

### closeConnection

This class closes the connection to the DB.

## driverHelper

This is the most important class of the whole framework. This class initializes the WebDriver / AppiumDriver / Remote Driver in order to execute the automated tests.

### getDriver

The getDriver class has been created in such way that it can work on web applications and on mobile applications. Provision has also been made to incorporate the driver into a grid using selenium grid 4.

#### WebDriver

Webdriver is used to open a selenium connection so that the automated test can start.

#### AppiumDriver

AppiumDriver is used to open a connection to the Appium Server which will then start an automated test on an emulator or mobile device

#### RemoteDriver

RemoteDriver is used to open a connection to the selenium grid.

### openWebsite

This class is called to open the website on which the automated test will be running.

### closeDriver

This class will close the Driver connection.

## reportManager

This class where the setup is for the Extent Reports. This different methods need to be called to create the report.

### reportSetup

This class should be called at the beginning of the test to set up the report which will be created with the tests.

### createReport

This class is called at the beginning of the testNG suite to create the report.

## waitHelper

This class has 2 methods which could be used to wait for an element / object.

1. waitForElement
2. elementWait

# Selenium Grid 4

The following should be done to incorporate the automation into Selenium Grid. In an ideal world this will be hosted on server. A tester would then connect to the server via the URL.

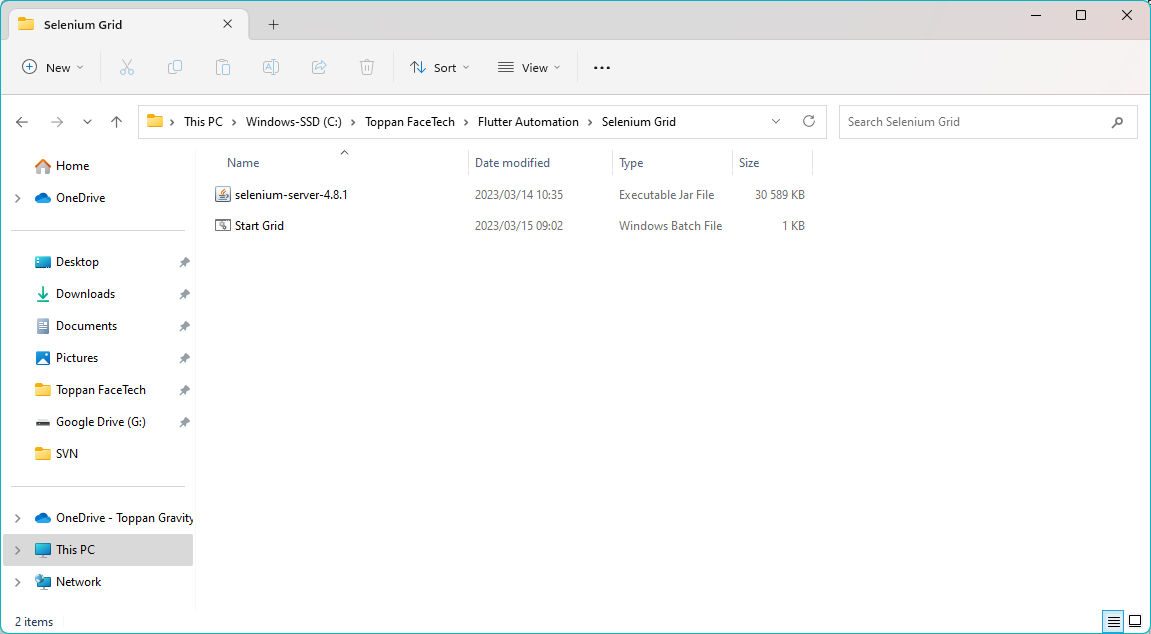
The current setup is on a local machine, but the same should be followed when hosting the Grid on a server.

## Download Jar file:

Download the Selenium Server jar from: <https://www.selenium.dev/downloads/>

## Create local folder:

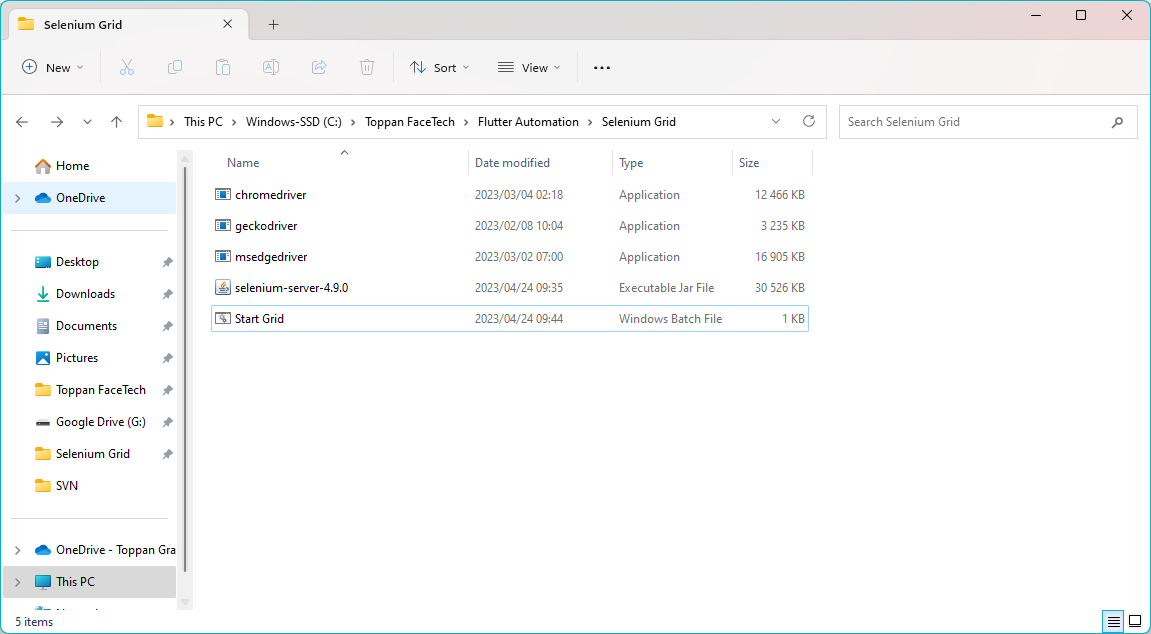
Create a local folder on the C/ D drive on your PC and add the Jar file to the created folder.



## Download browser drivers:

Download the following browser drivers from the websites and move the files to the above created folder.

* Chrome Driver 🡪 https://chromedriver.chromium.org/downloads
* Edge Driver 🡪 https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/
* Firefox Driver 🡪 https://github.com/mozilla/geckodriver/releases



## Grid Startup:

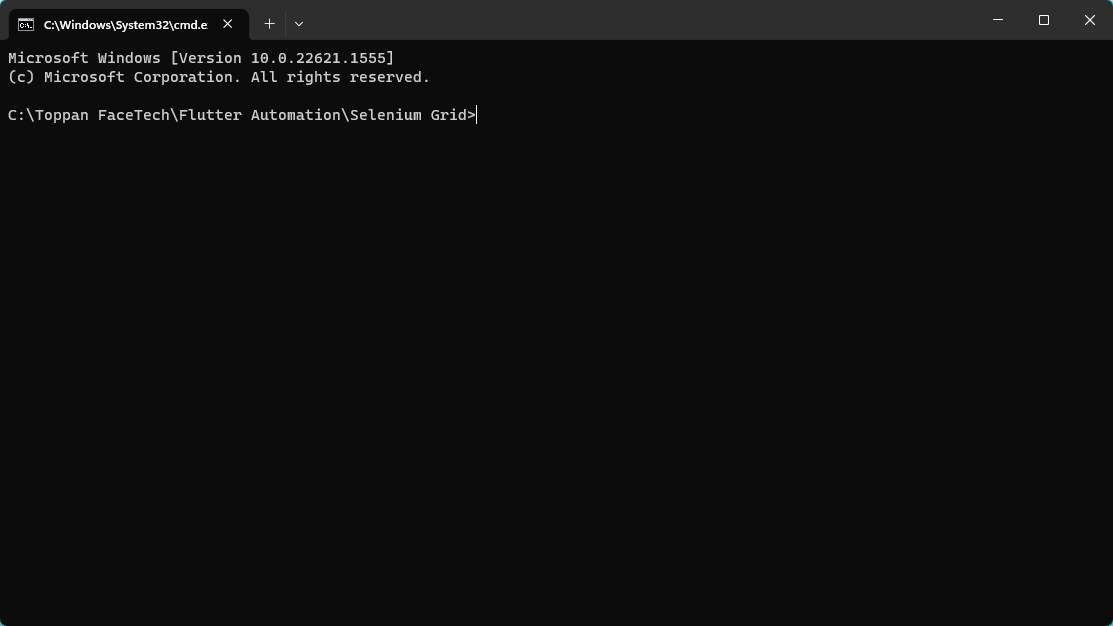
To start selenium grid as standalone the following command prompt should be used:

1. In the folder where that was created type cmd in the address bar and press enter:

Graphical user interface, text, application, email

Description automatically generated

1. The CMD terminal should open with the folder path:



1. Now enter the following command to start the server:

**java -jar selenium-server-4.9.0.jar standalone** and press enter.

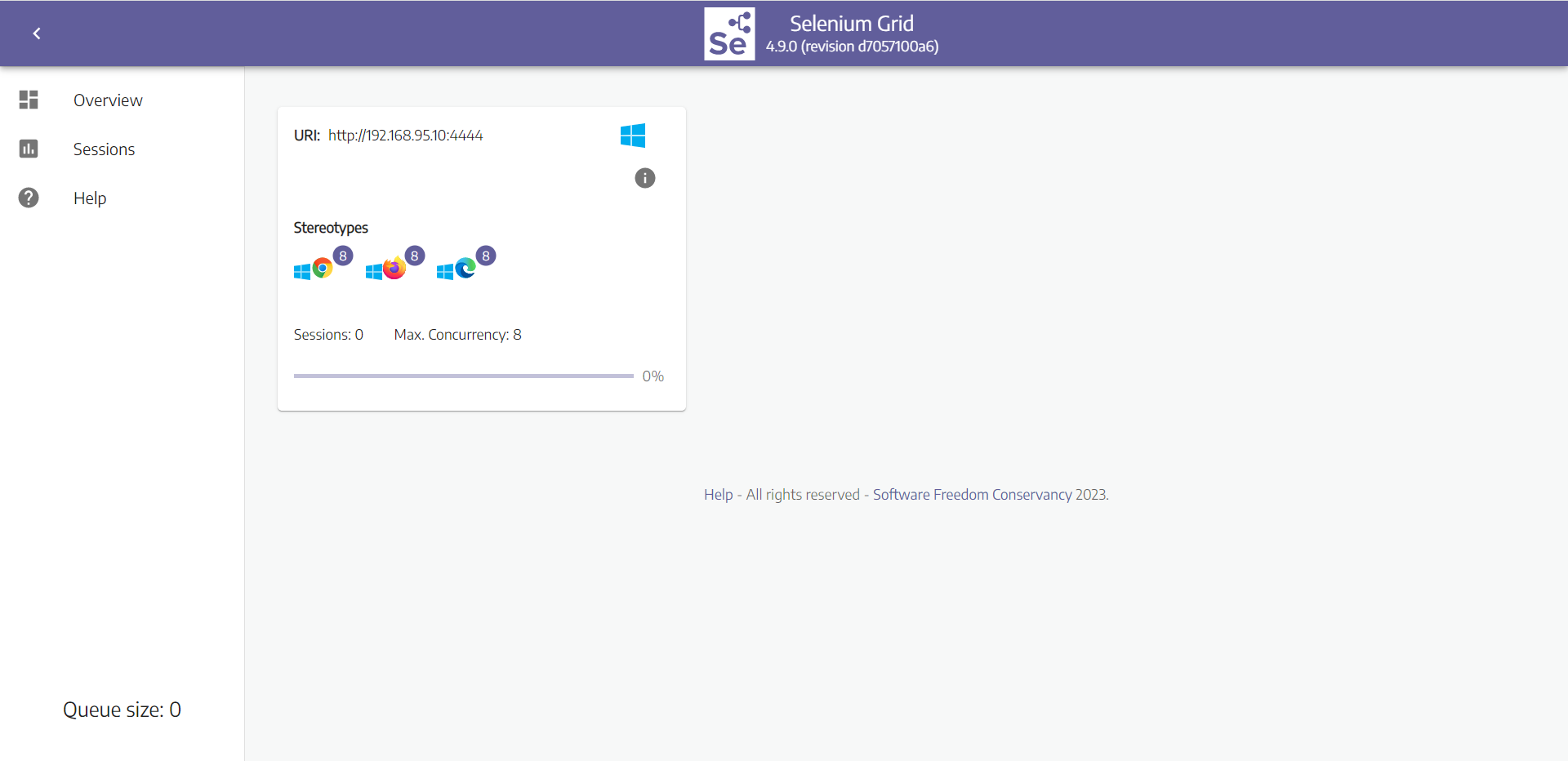
1. The server should start up.

Text

Description automatically generated

1. In your web browser enter the following address if working locally:

**localhost:4444** and the grid should open.



1. Create a .bat file with the command entered at step 3 and save the .bat file in the folder. This is handy to use when starting the Grid.