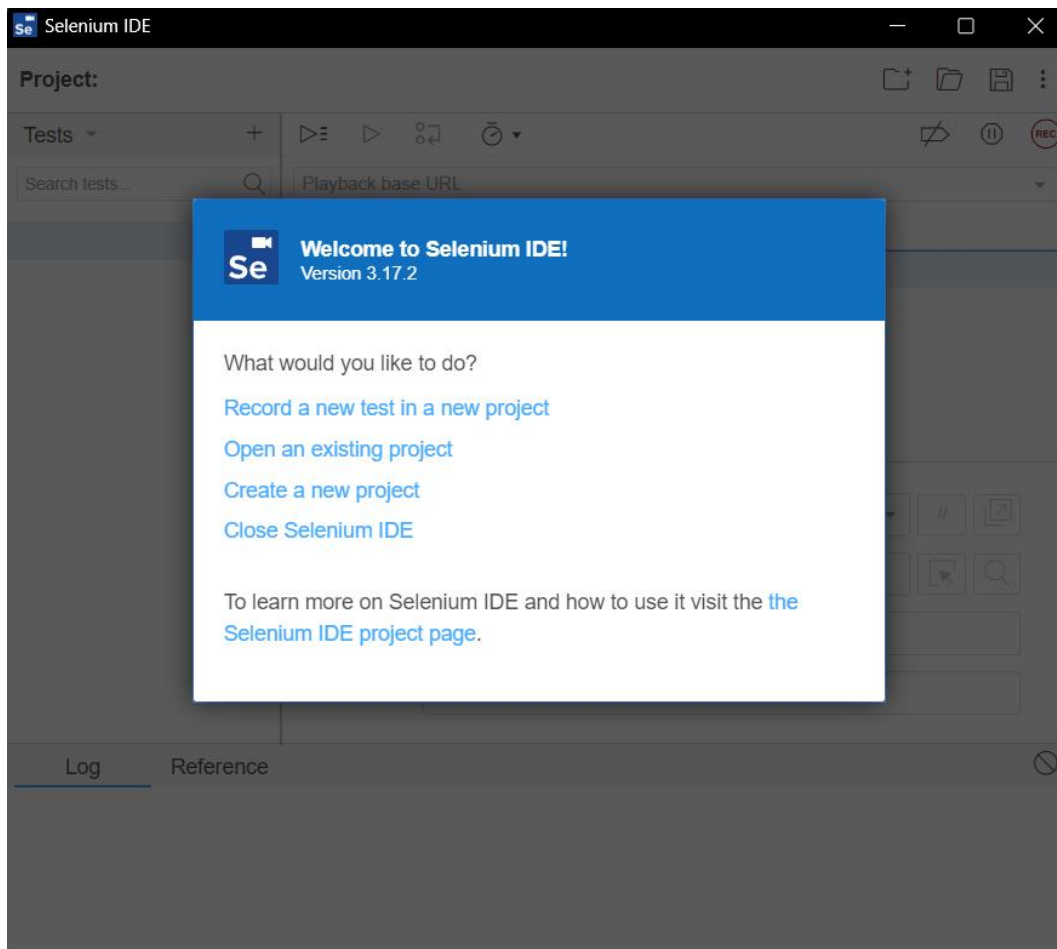


Selenium Training

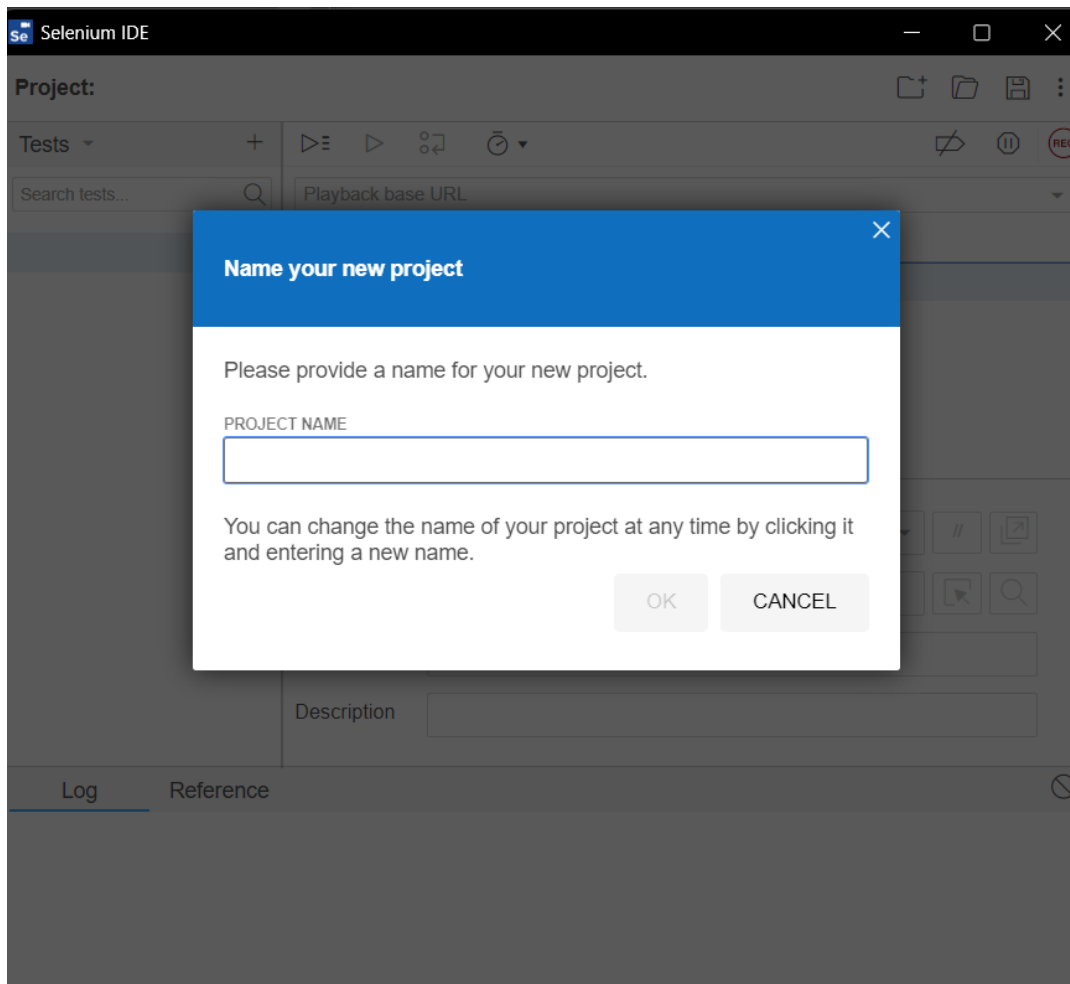
When creating a Selenium test it is fairly easy. To start you have two options when choosing how you want to approach testing. You can go the low code route and download the Selenium IDE web browser extension. The Selenium IDE is supported by most popular web browsers and can be easy to navigate compared to coding a Selenium test by hand. Or you can create a test using an IDE with Selenium WebDriver and N-Unit, where you can create tests in a robust language such as C#. There are various drivers to choose from to import such as the Selenium ChromeDriver for use with Google Chrome.

To start creating Selenium tests with the Selenium IDE extension first, download the extension in your browser. Once downloaded, open the Selenium IDE extension. You will come across a screen like the image below.

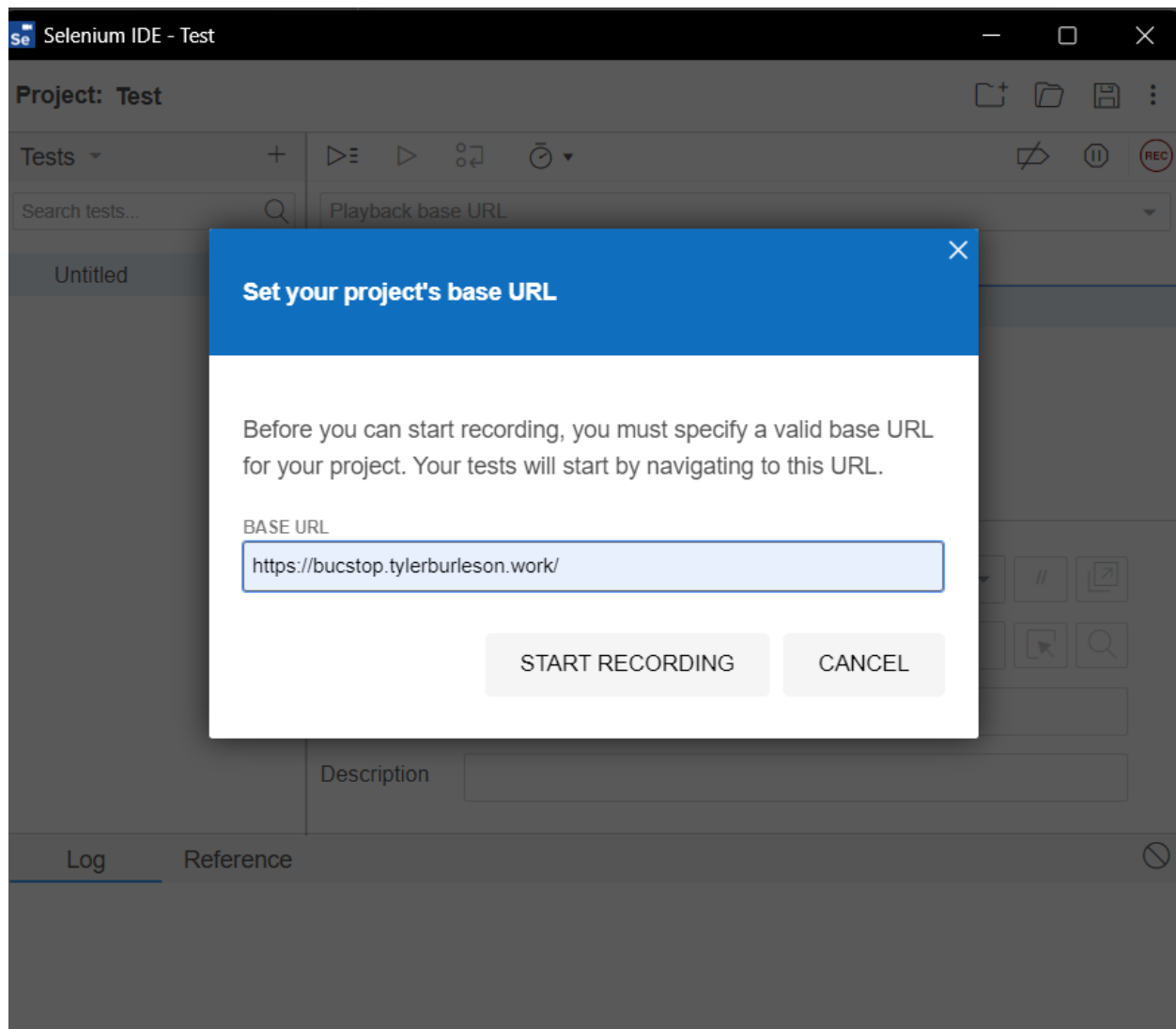


As you can see you have a few options: creating a new test in a new project, opening an existing project, creating a new project without any tests inside, and closing the IDE.

For the purpose of this walkthrough, we will choose to create a new test and a new project. Once clicked you will see a screen like the image below where you will be prompted to enter a name for your project.



Next you will need to enter a web address of the site that you wish to test within Selenium. Enter the address and click “Start Recording” to continue. You can see an example of this below.



After the web browser comes up showing the website you entered it will be recording at this moment until you close the web browser to end the test. You can select various buttons, links, or any web objects on the site to interact with them within your testing of the site. If you enter text into the fields of a site the input will be recorded as well.

Once you finished the test and closed the browser you will see the test in a list within your project. You can click the three dots beside the test name to access other options such as renaming and exporting the test to a .CS file to use within Visual Studio. Below is an example of this screen showing a range of tests I created in a project. When exporting for use in a C# project choose N-Unit C# as your exporting option.

Selenium IDE - BucStop

Project: BucStop

Tests

Search tests...

Admin Page Nav

FAQ Page Nav

Game Add Page Nav

Game Links Work

Game Page Nav

Privacy Page Nav

Student Login

Student Logout

Use Site as Guest

https://bucstop.tylerburleson.work

	Command	Target	Value
1	open	/	
2	set window size	818x864	
3	click	linkText=Admin	

Command

open

//

Target

/

Value

Description

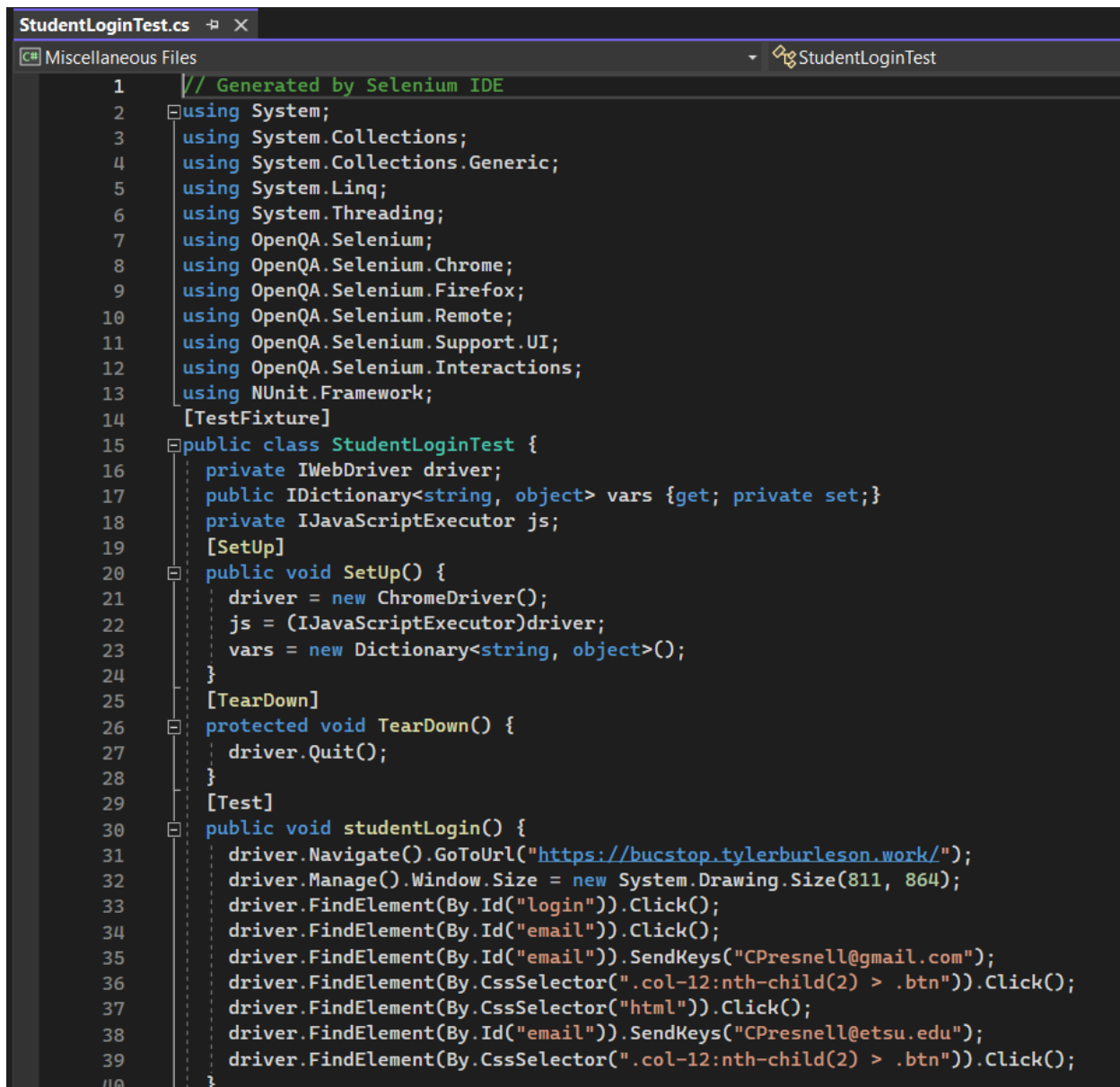
Log

Reference

To replay a test, you can choose the test and then click the play button. If you find that the test goes too fast for you, you can change the speed of the playback with the stopwatch icon. In the window you can see the actions the test is taking from the recording you had created; the IDE will show any test actions that have failed and any that have been done successfully.

To create a Selenium test using Visual Studio make sure you have imported a Webdriver for Selenium and also N-Unit within your IDE. I would recommend when creating the project create it as a N-Unit test project. Below you can see a test I created and Exported with Selenium IDE as a .CS file. This is one option or you can learn to use the Selenium WebDriver to perform actions using the

driver class, this essentially is the same as you had done previously when clicking on the website to perform actions. Once you have completed creating the test you can run it in Visual Studio with the test explorer, this will show you how the test performs and to see if it passes or fails.



```
1 // Generated by Selenium IDE
2 using System;
3 using System.Collections;
4 using System.Collections.Generic;
5 using System.Linq;
6 using System.Threading;
7 using OpenQA.Selenium;
8 using OpenQA.Selenium.Chrome;
9 using OpenQA.Selenium.Firefox;
10 using OpenQA.Selenium.Remote;
11 using OpenQA.Selenium.Support.UI;
12 using OpenQA.Selenium.Interactions;
13 using NUnit.Framework;
14 [TestFixture]
15 public class StudentLoginTest {
16     private IWebDriver driver;
17     public IDictionary<string, object> vars {get; private set;}
18     private IJavaScriptExecutor js;
19     [SetUp]
20     public void Setup() {
21         driver = new ChromeDriver();
22         js = (IJavaScriptExecutor)driver;
23         vars = new Dictionary<string, object>();
24     }
25     [TearDown]
26     protected void TearDown() {
27         driver.Quit();
28     }
29     [Test]
30     public void studentLogin() {
31         driver.Navigate().GoToUrl("https://bucstop.tylerburleson.work/");
32         driver.Manage().Window.Size = new System.Drawing.Size(811, 864);
33         driver.FindElement(By.Id("login")).Click();
34         driver.FindElement(By.Id("email")).Click();
35         driver.FindElement(By.Id("email")).SendKeys("CPresnell@gmail.com");
36         driver.FindElement(By.CssSelector(".col-12:nth-child(2) > .btn")).Click();
37         driver.FindElement(By.CssSelector("html")).Click();
38         driver.FindElement(By.Id("email")).SendKeys("CPresnell@etsu.edu");
39         driver.FindElement(By.CssSelector(".col-12:nth-child(2) > .btn")).Click();
40     }
```

I would recommend watching YouTube tutorials or find a site with a Selenium WebDriver syntax walkthrough to learn more about what actions can be performed using the driver class. Also, don't forget to perform an exit action with the driver so the test will end if it fails. Without doing so the test will continue to run until you stop it in Visual Studio.