SELENIUM WEBDRIVER

EXERCISE BOOK

**Software Developer Level 4: Module Three**

**Contents**

[Exercise 1 – Google search for kittens 4](#_Toc124930793)

[Exercise 2 – Check the chapter titles 4](#_Toc124930794)

[Exercise 3 – SwagLabs SauceDemo 5](#_Toc124930795)

[a) Working account 5](#_Toc124930796)

[b) Locked out account 5](#_Toc124930797)

[c) Incorrect password account 5](#_Toc124930798)

[Exercise 4 – Creating a shopping cart and placing an order 7](#_Toc124930799)

[a) Adding items to the cart 7](#_Toc124930800)

[b) Placing an order 7](#_Toc124930801)

[Exercise 5 – Google search form (test\_site > Exercises > Forms) 9](#_Toc124930802)

[Task 1 9](#_Toc124930803)

[Task 2 9](#_Toc124930804)

[Task 3 9](#_Toc124930805)

[Exercise 6 – Login form (test\_site > Exercises > Forms) 10](#_Toc124930806)

[Task 1 10](#_Toc124930807)

[Task 2 10](#_Toc124930808)

[Exercise 7 – Editable users table (test\_site > Exercises > Tables) 11](#_Toc124930809)

[Task 1) Deleting users 11](#_Toc124930810)

[Task 2) Editing users 11](#_Toc124930811)

[Task 3) Cancelling a user edit (part 1) 11](#_Toc124930812)

[Task 4) Cancelling a user edit (part 2) 11](#_Toc124930813)

[Exercise 8 – Drag and drop text (test\_site > Exercises > Drag and drop) 12](#_Toc124930814)

[Task: Drag and drop the text 12](#_Toc124930815)

[Exercise 9 – Drag and drop cards between columns (test\_site > Exercises > Drag and drop) 13](#_Toc124930816)

[Task 1: From Todo to Doing 13](#_Toc124930817)

[Task 2: From Doing to Done 13](#_Toc124930818)

[Exercise 10 – Waiting for elements to load (test\_site > Exercises > Waits) 14](#_Toc124930819)

[Task 1: Verify the elements load within 10 seconds 14](#_Toc124930820)

[Task 2: Verify the spinning loader appears and disappears within 10 seconds 14](#_Toc124930821)

[Exercise 11 – Page Object Model 15](#_Toc124930822)

[Task 1: Home page 15](#_Toc124930823)

[Task 2: Exercises page 15](#_Toc124930824)

[Task 3: Exercise category pages 15](#_Toc124930825)

[Task 4: Tests 15](#_Toc124930826)

# Exercise 1 – Google search for kittens

Using Selenium WebDriver, create a test case which:

1. Opens the **google.com** web page
2. Searches for **kittens**
3. Verifies the text in the tab (title) is **kittens – Google Search**

Hints:

* You will need to use the **title** assertion.
* You will need to accept cookies.

After creating the test case, run it to verify it works correctly. Inspect the generated commands to familiarise yourself with them.

# Exercise 2 – Check the chapter titles

For this exercise, you will be using [**https://automatetheboringstuff.com**](https://automatetheboringstuff.com). Your task is to open three chapters and assert the pages main heading, returning to the previous page and repeating for each chapter .

Hints:

To **WebDriver** API has a **Navigation** object which you can use

# Exercise 3 – SwagLabs SauceDemo

SwagLabs created a site called **SauceDemo**, this can be used for some simple user-acceptance tests. Your task will be to test a variety of logins, supplied below, and asserting that the result is expected.

**URL**: <https://www.saucedemo.com>

**Working account:**

Username: standard\_user

Password: secret\_sauce

**Locked out account**:

Username: locked\_out\_user

Password: secret\_sauce

**Incorrect password account**:

Username: problem\_user

Password: wrong

For each of the following exercises, create an associated test case.

## Working account

Using the supplied account, login the user to the SauceDemo page. Once logged in, verify this by asserting that a product is present.

## Locked out account

Using the supplied account, attempt to login the locked-out user. After attempting this, you will receive an error message:

* Epic sadface: Sorry, this user has been locked out.

Assert that this error message does in-fact show.

## Incorrect password account

Using the supplied account, attempt to login the user with the incorrect password. After attempting this, you will receive the error message:

* Epic sadface: Username and password do not match any user in this service

Assert that this error message does in-fact show.

# Exercise 4 – Creating a shopping cart and placing an order

The following exercises use a demo product store.

**URL**: <https://www.demoblaze.com/>

## Adding items to the cart

Create a Selenium WebDriver test case which adds at least three items to the shopping basket.

* This site is a demo site, so please don’t worry about any actual purchases

After adding three items to the basket, navigate to the **Cart** and assert that the items were in fact added to the basket.

## Placing an order

Create a Selenium WebDriver test case which adds at least one item to the basket. After adding the items to the basket, automate the process of checking out. Automating the checkout process will involve clicking the **Place Order** button, filling out the form (with fake details) and selecting **Purchase**.

After selecting **Purchase**, a pop-up will appear:

Graphical user interface, text, application, chat or text message

Description automatically generated

Assert that the **Amount**, **Card Number** and **Name** are as expected.

# Exercise 5 – Google search form (test\_site > Exercises > Forms)

This is a forms exercise which can be found on the supplied test website in the labs folder, called **test\_site.jar**. Instructions for running this are in the **README.md** in the same folder.

## Task 1

Create a Selenium test that verifies the search functionality works. Things to check for in your assertions could include:

* The pages title.
* The content in the Google search bar after pressing submit.

## Task 2

Create a Selenium test that verifies the clear search content functionality works. Things to check for in your assertions could include:

* The search bar having no input value present.

## Task

Create a Selenium test that verifies the HTML5 form validation works, in this case that a search value is required.

# Exercise 6 – Login form (test\_site > Exercises > Forms)

This is a forms exercise which can be found on the supplied test website in the labs folder, called **test\_site.jar**. Instructions for running this are in the **README.md** in the same folder.

## Task 1

Your task is to use Selenium to validate that the correct messages appear for correct and incorrect login details. The username is "admin" and the password is "password", any other details are invalid.

The messages to check for include:

* **Successful login**: Welcome back to your portal admin!
* **Failed login**: Incorrect login details, please try again!

## Task 2

Use Selenium to validate that the logout functionality is working correctly. After logging out, the following message will be temporarily displayed for 10 seconds:

* Logged out successfully!

Verify the message does appear and then disappear after 10 seconds

# Exercise 7 – Editable users table (test\_site > Exercises > Tables)

This is a tables exercise which can be found on the supplied test website in the labs folder, called **test\_site.jar**. Instructions for running this are in the **README.md** in the same folder.

## Task 1) Deleting users

Create a Selenium test which verifies that a deleted user is removed from the DOM.

## Task 2) Editing users

Create a Selenium test which verifies that changes made to a users details are saved to the DOM when the 'Save' button is pressed. To make the 'Save' button appear, you first have to press the 'Edit' button.

## Task 3) Cancelling a user edit (part 1)

Create a Selenium test which verifies that changes made to a users details are not saved to the DOM when the 'Cancel' button is pressed. To make the 'Cancel' button appear, you first have to press the 'Edit' button.

## Task 4) Cancelling a user edit (part 2)

Create a Selenium test which verifies that changes made to a users details are not saved to the DOM when the 'Edit' button of another user is selected.

# Exercise 8 – Drag and drop text (test\_site > Exercises > Drag and drop)

This is a drag and drop exercise which can be found on the supplied test website in the labs folder, called **test\_site.jar**. Instructions for running this are in the **README.md** in the same folder.

WebDriver has a bug when working with the browsers drag and drop functionality, to compensate for this the Selenium developers recommend executing a portion of JavaScript to manually trigger the drag and drop events in the browser. A method containing this JavaScript is available in the **labs/drag\_and\_drop.md** file.

## Task: Drag and drop the text

For this task, you must use Selenium to drag and drop the text into the square shaped div element. Ensure you verify that the element is moved inside of the square drop target in your Selenium test using an assertion.

# Exercise 9 – Drag and drop cards between columns (test\_site > Exercises > Drag and drop)

This is a drag and drop exercise which can be found on the supplied test website in the labs folder, called **test\_site.jar**. Instructions for running this are in the **README.md** in the same folder.

*WebDriver has a bug when working with the browsers drag and drop functionality, to compensate for this the Selenium developers recommend executing a portion of JavaScript to manually trigger the drag and drop events in the browser. A method containing this JavaScript is available in the* ***labs/drag\_and\_drop.md*** *file.*

## Task 1: From Todo to Doing

Create a Selenium test which verifies that a card can be moved from the 'Todo' column to the 'Doing' column. You will need to verify at least two different things, one of them being that the card does appear in the DOM in the column it was dropped in.

## Task 2: From Doing to Done

Create a Selenium test which verifies that the 'Plan weekly party' card in the 'Doing' column can be moved to the 'Done' column. As before, there are at least two things you should assert in your test.

# Exercise 10 – Waiting for elements to load (test\_site > Exercises > Waits)

This is a waits exercise which can be found on the supplied test website in the labs folder, called **test\_site.jar**. Instructions for running this are in the **README.md** in the same folder.

* *1 second is 1000 milliseconds (1s = 1000ms)*

## Task 1: Verify the elements load within 10 seconds

Use Selenium to verify that the elements load in within 10 seconds. There are two elements to check for. A h1 element with an id of wait-target-1 and an adjacent p element.

You must use an explicit wait for this task.

## Task 2: Verify the spinning loader appears and disappears within 10 seconds

Use Selenium to verify that the spinning loader element appears when the page opens, and disappears within 10 seconds. You should also verify that the loader is replaced with the expected content.

The loader is a div with an id of ex1-loader. You should also take two screenshots of the page, one demonstrating a visible loader and one demonstrating the loaded content. Save the screenshots to your filesystem.

# Exercise 11 – Page Object Model

For this exercise, you are required to create page object models which represent the supplied test site.

*You may wish to look at the tests (task 4) first, and then build out your POM classes features as you do each test.*

## Task 1: Home page

Create a Page Object Model of the home page.

## Task 2: Exercises page

Create a Page Object Model of the exercises page.

## Task 3: Exercise category pages

Create a Page Object Model for each of the exercise category pages.

## Task 4: Tests

Using your POM classes, perform the following tests:

1. Assert the home pages main h1 text.
2. Assert the exercises page main h1 text.
3. Assert the categories are present on the exercise page.
4. Assert that each category link on the exercise page works.
5. Rewrite your solutions for exercises 5 – 10 to use the POM design pattern