East West University

Lab Report 03

Topic: Selenium IDE

Course Title: Software Testing and Quality Assurance

Course Code: CSE 430

Section No: 01

Submitted By

Md. Saiful Islam

2022-3-60-045

Submitted To

Dr. Shamim H Ripon

Professor

Department of Computer Science and Engineering

East West University

Task 1: Wikipedia Search Test

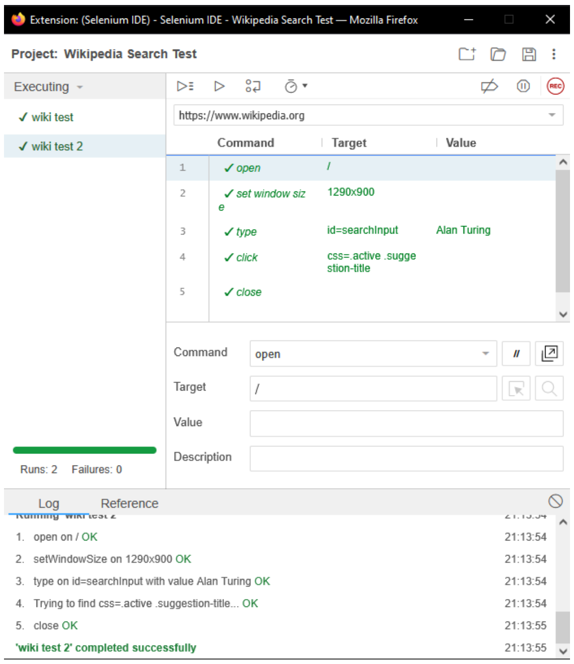
Objective: Verify search functionality and page redirection.

Target URL: https://www.wikipedia.org/

Steps:

- **open**: Navigate to the base URL.
- **type**: Enter "Alan Turing" into the search input field.
- **click**: Click the search button.
- **assertTitle**: Verify the window title matches "Alan Turing - Wikipedia".

Output:


Extension: (Selenium IDE) - Selenium IDE - Wikipedia Search Test — Mozilla Firefox
Project: Wikipedia Search Test
Executing
✓ wiki test
✓ wiki test 2
Runs: 2 Failures: 0
Log
Reference

https://www.wikipedia.org

	Command	Target	Value
1	✓ open	/	
2	✓ set window size	1290x900	
3	✓ type	id=searchInput	Alan Turing
4	✓ click	css=.active .suggestion-title	
5	✓ close		

Command	open
Target	/
Value	
Description	

```
1. open on / OK                                                     21:13:54
2. setWindowSize on 1290x900 OK                                     21:13:54
3. type on id=searchInput with value Alan Turing OK                 21:13:54
4. Trying to find css=.active .suggestion-title... OK               21:13:54
5. close OK                                                         21:13:55
'wiki test 2' completed successfully                                21:13:55
```

Reflection: This task demonstrated how Selenium IDE can automate basic search functionality and verify correct page redirection. By validating the page title after searching, we confirmed that the application responds accurately to user input and navigates to the expected result page.

Task 2: Dropdown Test on Booking.com

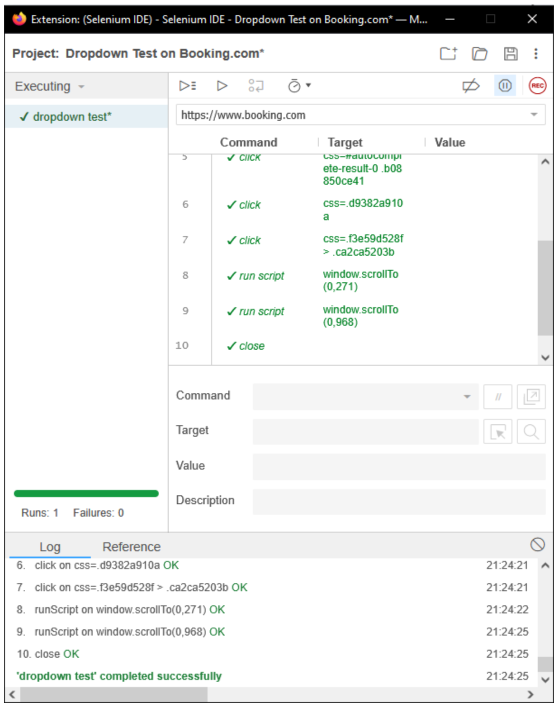
Objective: Automate location search and dropdown selection.

Target URL: https://www.booking.com/

Steps:

- **open**: Navigate to Booking.com.
- **type**: Enter "Dhaka" into the destination search box.
- **waitForElementVisible**: Wait for the autocomplete suggestions to appear.
- **click**: Select the first suggestion from the dropdown.
- **assertText**: Verify the selected location matches the input.

Output:


Extension: (Selenium IDE) - Selenium IDE - Dropdown Test on Booking.com* — M...

Project: Dropdown Test on Booking.com*

Executing

✓ dropdown test*

https://www.booking.com

	Command	Target	Value
5	✓ click	css=#autocomplete-result-0 .b08850ce41	
6	✓ click	css=.d9382a910a	
7	✓ click	css=.f3e59d528f > .ca2ca5203b	
8	✓ run script	window.scrollTo(0,271)	
9	✓ run script	window.scrollTo(0,968)	
10	✓ close		

Command
Target
Value
Description

Runs: 1 Failures: 0

Log Reference

```
6.  click on css=.d9382a910a OK                          21:24:21
7.  click on css=.f3e59d528f > .ca2ca5203b OK            21:24:21
8.  runScript on window.scrollTo(0,271) OK               21:24:22
9.  runScript on window.scrollTo(0,968) OK               21:24:25
10. close OK                                             21:24:25
'dropdown test' completed successfully                   21:24:25
```

Reflection: Through this task, we learned how to handle dynamic dropdown elements and autocomplete suggestions using Selenium IDE. It helped us understand synchronization issues and the importance of waiting for elements to appear before interaction.

Task 3: Product Add-to-Cart Test

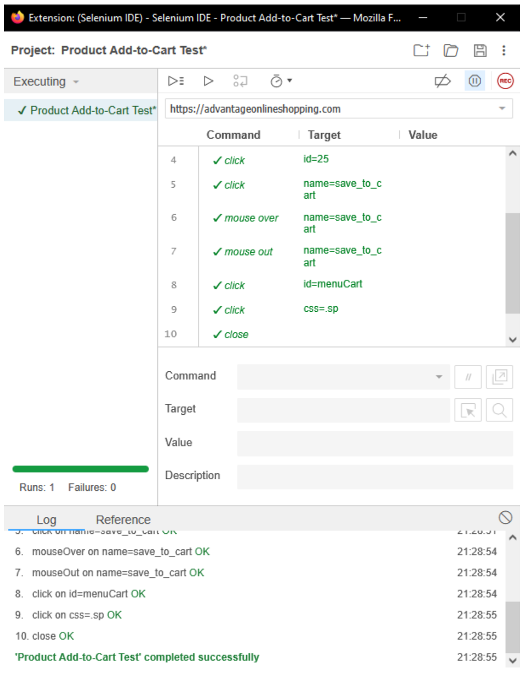
Objective: Validate the "Add to Cart" workflow on an e-commerce site.

Target URL: https://advantageonlineshopping.com

Steps:

- **open**: Navigate to the homepage.
- **click**: Select a product category (e.g., Speakers).
- **click**: Choose a specific product.
- **click**: Click the "Add to Cart" button.
- **assertText**: Verify the confirmation message or check that the cart count has updated.

Output:


Extension: (Selenium IDE) - Selenium IDE - Product Add-to-Cart Test* — Mozilla F...

Project: Product Add-to-Cart Test*

Executing

✓ Product Add-to-Cart Test*

https://advantageonlineshopping.com

	Command	Target	Value
4	✓ click	id=25	
5	✓ click	name=save_to_cart	
6	✓ mouse over	name=save_to_cart	
7	✓ mouse out	name=save_to_cart	
8	✓ click	id=menuCart	
9	✓ click	css=.sp	
10	✓ close		

Runs: 1   Failures: 0

```
6.  mouseOver on name=save_to_cart OK            21:28:54
7.  mouseOut on name=save_to_cart OK             21:28:54
8.  click on id=menuCart OK                      21:28:54
9.  click on css=.sp OK                          21:28:55
10. close OK                                     21:28:55
'Product Add-to-Cart Test' completed successfully 21:28:55
```

Reflection: This task provided experience in automating a common e-commerce workflow. We verified that product selection and cart updates function correctly, reinforcing the importance of validating user actions that affect application state.

Task 4: Negative Login Validation

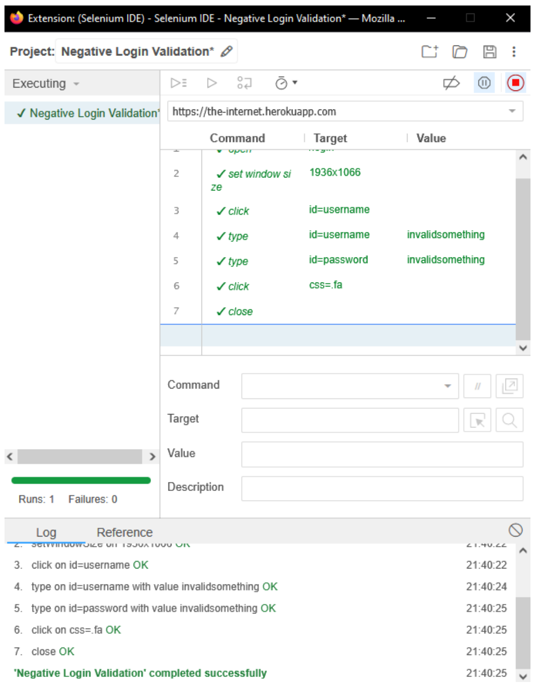
Objective: Verify error messages when using invalid credentials.

Target URL: https://the-internet.herokuapp.com/login

Steps:

1. **open**: Navigate to the login page.
2. **type**: Enter an invalid username (e.g., "wrongUser").
3. **type**: Enter an invalid password (e.g., "wrongPass").
4. **click**: Click the "Login" button.
5. **assertText**: Verify the error message contains "Your username is invalid!".

Output:


Extension: (Selenium IDE) - Selenium IDE - Negative Login Validation* — Mozilla ...
Project: Negative Login Validation*
Executing
✓ Negative Login Validation*
Log
Reference
Runs: 1 Failures: 0

https://the-internet.herokuapp.com

	Command	Target	Value
2	✓ set window size	1936x1066	
3	✓ click	id=username	
4	✓ type	id=username	invalidsomething
5	✓ type	id=password	invalidsomething
6	✓ click	css=.fa	
7	✓ close		

Command
Target
Value
Description

```
3.  click on id=username OK                                    21:40:22
4.  type on id=username with value invalidsomething OK         21:40:24
5.  type on id=password with value invalidsomething OK         21:40:25
6.  click on css=.fa OK                                        21:40:25
7.  close OK                                                   21:40:25
'Negative Login Validation' completed successfully             21:40:25
```

Reflection: This task focused on negative testing by using invalid login credentials. It highlighted the importance of validating error messages and ensuring that the system provides appropriate feedback when authentication fails.

Task 5: Form Submission (DemoQA)

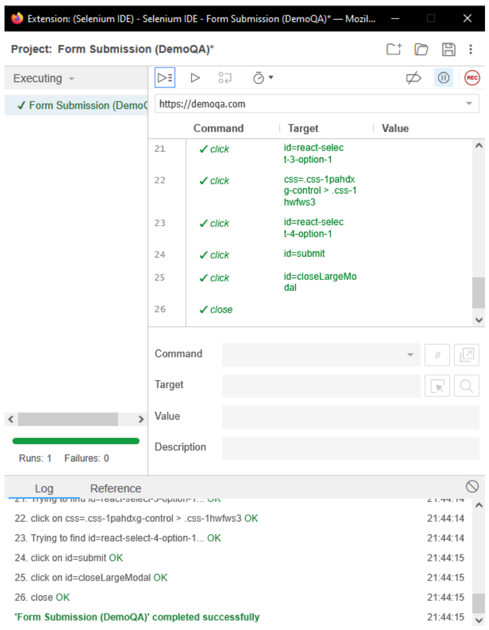
Objective: Automate filling and submitting a comprehensive web form.

Target URL: https://demoqa.com/automation-practice-form

Steps:

1. **open**: Navigate to the form page.
2. **type**: Fill in the First Name, Last Name, and Email fields.
3. **click**: Select a Gender radio button and enter a Mobile Number.
4. **click**: Submit the form (Note: You may need to scroll down or zoom out if the submit button is obscured).
5. **verifyElementPresent**: Verify the "Thanks for submitting the form" confirmation modal appears.

Output:


Extension: (Selenium IDE) - Selenium IDE - Form Submission (DemoQA)* — Mozil...
Project: Form Submission (DemoQA)*
Executing
✓ Form Submission (DemoQ
https://demoqa.com
Runs: 1 Failures: 0
Command
Target
Value
Description
Log
Reference

	Command	Target	Value
21	✓ click	id=react-select-3-option-1	
22	✓ click	css=.css-1pahdxg-control > .css-1hwfws3	
23	✓ click	id=react-select-4-option-1	
24	✓ click	id=submit	
25	✓ click	id=closeLargeModal	
26	✓ close		

```
22. click on css=.css-1pahdxg-control > .css-1hwfws3 OK        21:44:14
23. Trying to find id=react-select-4-option-1... OK            21:44:14
24. click on id=submit OK                                      21:44:15
25. click on id=closeLargeModal OK                             21:44:15
26. close OK                                                   21:44:15
'Form Submission (DemoQA)' completed successfully              21:44:15
```

Reflection: By automating a detailed form submission, we practiced interacting with multiple input types within a single test case. This task emphasized data entry validation and confirmation message verification in form-based applications.

Task 6: Responsive Navigation (W3Schools)

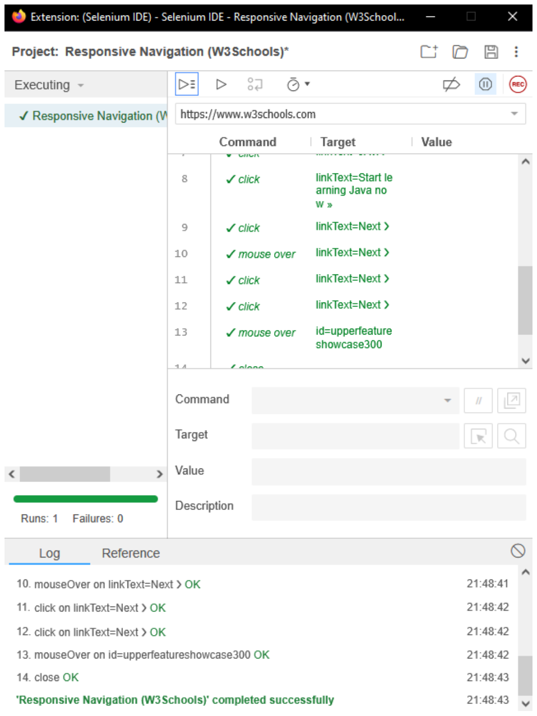
Objective: Test navigation menus on a responsive website.

Target URL: https://www.w3schools.com/

Steps:

1. **open**: Navigate to the homepage.
2. **click**: Click the responsive menu icon (hamburger menu) or "Tutorials".
3. **click**: Select "Learn JavaScript" from the menu.
4. **assertText**: Verify that the heading "JavaScript Tutorial" or "JavaScript Examples" is visible on the target page.

Output:



Extension: (Selenium IDE) - Selenium IDE - Responsive Navigation (W3School...

Project: Responsive Navigation (W3Schools)*

Executing

✓ Responsive Navigation (W

https://www.w3schools.com

	Command	Target	Value
8	✓ click	linkText=Start learning Java now »	
9	✓ click	linkText=Next ❯	
10	✓ mouse over	linkText=Next ❯	
11	✓ click	linkText=Next ❯	
12	✓ click	linkText=Next ❯	
13	✓ mouse over	id=upperfeatureshowcase300	

Command

Target

Value

Description

Runs: 1 Failures: 0

Log Reference

```
10. mouseOver on linkText=Next ❯ OK                          21:48:41
11. click on linkText=Next ❯ OK                              21:48:42
12. click on linkText=Next ❯ OK                              21:48:42
13. mouseOver on id=upperfeatureshowcase300 OK               21:48:42
14. close OK                                                 21:48:43
'Responsive Navigation (W3Schools)' completed successfully   21:48:43
```

Reflection: This task helped us understand how to test navigation behavior on responsive websites. We verified that menu interactions lead to the correct content, ensuring usability across different layouts and screen conditions.

Task 7: File Upload Automation

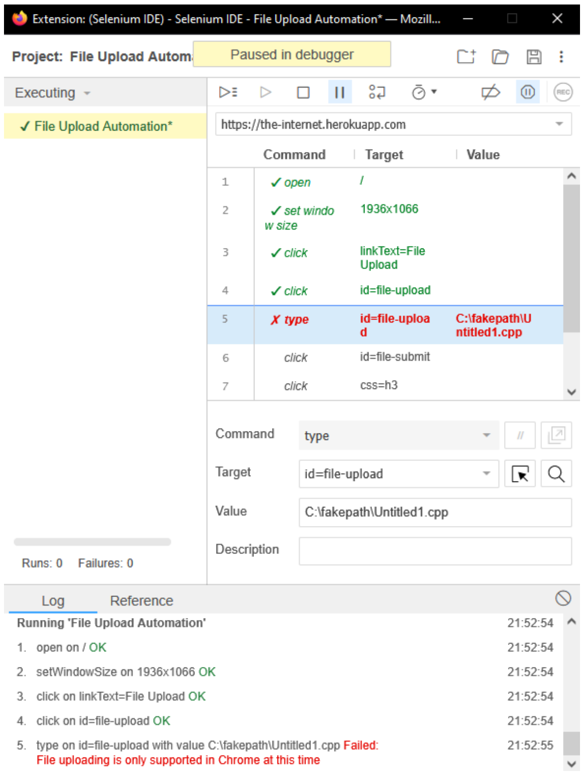
Objective: Automate the file upload process and validate success.

Target URL: https://the-internet.herokuapp.com/upload

Steps:

1. **open**: Navigate to the upload page.
2. **type**: Set the file input target to a local file path (e.g., a sample .txt file).
3. **click**: Click the "Upload" button.
4. **assertText**: Verify the header text reads "File Uploaded!".

Output:



Extension: (Selenium IDE) - Selenium IDE - File Upload Automation* — Mozill...

Project: File Upload Autom

Paused in debugger

Executing

✓ File Upload Automation*

https://the-internet.herokuapp.com

	Command	Target	Value
1	✓ open	/	
2	✓ set window size	1936x1066	
3	✓ click	linkText=File Upload	
4	✓ click	id=file-upload	
5	✗ type	id=file-upload	C:\fakepath\Untitled1.cpp
6	click	id=file-submit	
7	click	css=h3	

Command	type
Target	id=file-upload
Value	C:\fakepath\Untitled1.cpp
Description	

Runs: 0 Failures: 0

Log Reference

```
Running 'File Upload Automation'                                      21:52:54
1. open on / OK                                                       21:52:54
2. setWindowSize on 1936x1066 OK                                      21:52:54
3. click on linkText=File Upload OK                                   21:52:54
4. click on id=file-upload OK                                         21:52:54
5. type on id=file-upload with value C:\fakepath\Untitled1.cpp Failed:   21:52:55
   File uploading is only supported in Chrome at this time
```

Reflection: This task demonstrated how Selenium IDE handles file input elements and validates upload success. It reinforced the importance of testing file handling features that rely on user-provided local resources. In our Case the File upload failed as File uploading is only supported in Chrome at this time.

Conclusion

This lab provided practical experience in using **Selenium IDE** to automate and validate common web application functionalities. Through various test cases such as search operations, dropdown handling, add-to-cart workflows, negative login validation, form submission, responsive navigation, and file upload automation, we learned how to create reliable and repeatable test scripts. The lab enhanced our understanding of automated functional testing, synchronization techniques, and assertion-based validation. Overall, this experiment demonstrated how Selenium IDE can significantly reduce manual testing effort while improving accuracy, efficiency, and test coverage in web application testing.