

## Project 3: Graphical User Interface Testing Project


### CSE 565: Software Verification and Validation

#### Description of the Application

In this Graphical User Interface (GUI) testing project, I created a Website with the GUI application. The application has two versions, and each version has three pages: one for signing up, one for logging in, and one for the to-do list. Each page has things like pictures, places to write text, lists, sliding bars, labels, checkboxes, and buttons. Below I will explain these things more.


#### First Version

The signup page is used to collect all the credentials for signing someone up to this Website. This page has a picture and several boxes where I can type in personal details like name, email, password, and phone number. I also used little icons next to each box to show what kind of information to put in there, like an icon of a person for the name box. There is also a button I can click to choose gender (male or female), and there are buttons at the bottom that say Submit and Reset button (Figure 1). If I already have an account, there is a link that takes me to the Login page so I can log in.




Sign Up


Name




Email




Username




Password



Confirm Password



Phone Number



Gender

☐ Male ☐ Female

Submit

Reset

Already have an account? [Login](#)

Figure 1: Screenshot of signup page (Version 1)

Figure 2 is the Login page where there are two boxes. In those boxes I can type my username and password, and a little box I can check if I want the website to remember my login information. There's also a button I can click to send my information. Just like the "Signup" page, there's a link in this page that takes me to the "Signup" page if I don't have an

account yet.

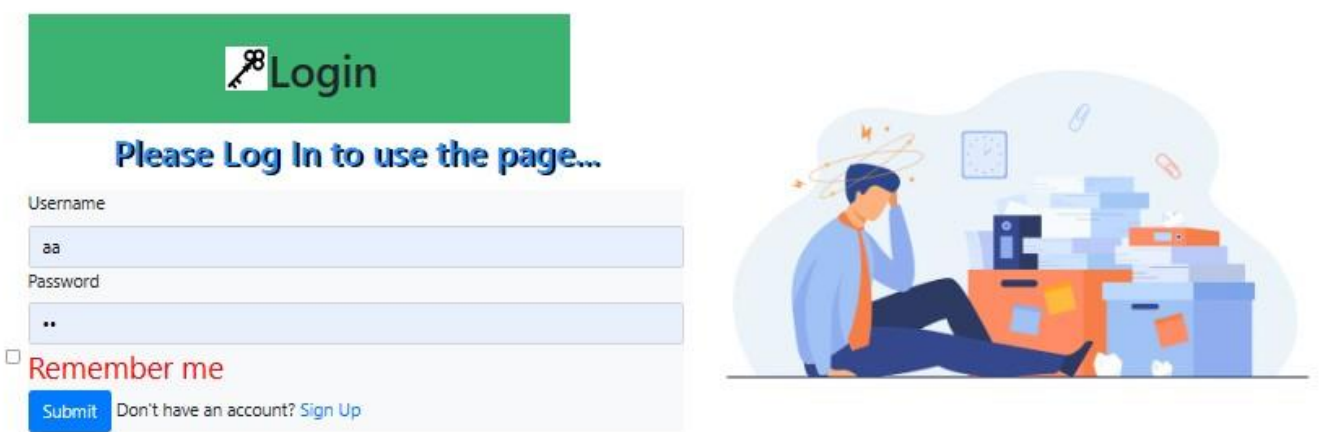


Figure 2: Screenshot of login page (Version 1)

On the To Do List page there's a picture and a box where I can write down my tasks. I also added some buttons I can click to say how important each task is, like "Low," "Medium," or "High" priority (see Figure 3). If I want to log out of my account, there's a button in the top left corner that says "Sign Out," and it takes me back to the "Login" page. This way, I can switch to a different account if I need to.

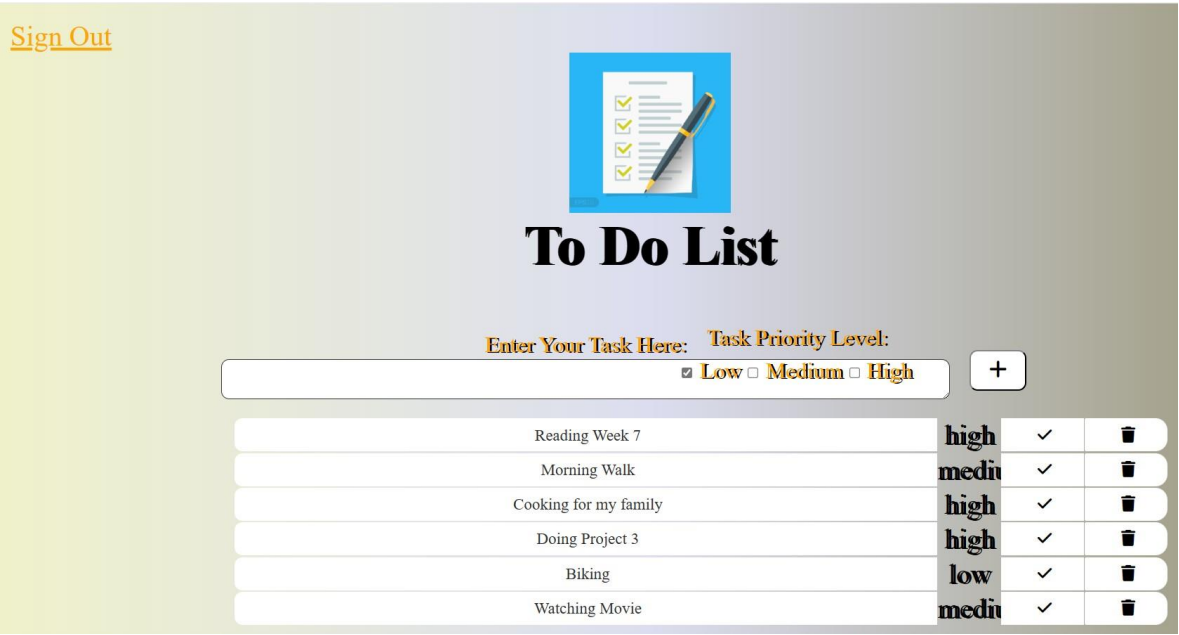



Figure 3: Screenshot of Todo list page (Version 1)

**Second Version**

In this version, I have changed some GUI elements in all the pages. These changes involve moving and resizing text boxes, images, buttons, and more. I've also adjusted how I navigate between the pages. Let me describe the modifications I made to each page in more detail.

In the Figure below, I can see the changes I made to the first page in the second version. I moved the Submit and Reset buttons to different places on the page. I also changed where the link to the login page is located and how it works. It's now situated differently, and the way I use it has been updated right at the bottom of the page close to the middle



**Sign Up**

Name

Email

Username

Password

Confirm Password

Phone Number

Gender  
☐ Male ☐ Female

Already have an account? [Login](#)

Figure 4: Screenshot of Signup page (version 2)

For the login page of this version, I've also made some changes which are shown in the figure below. In the Figure below, I will notice that I moved the image to a different spot on the page. It's now on the right side, and the login forms (like the username and password) have switched places with it. I made the key image on the left side bigger as Ill.

To make it easier for users to switch between pages, I updated how "Sign Up" link works. Now, when I click on it, it takes me directly to the "To Do List" page.



Figure 5: Screenshot of login page (version 2)

I've made three significant changes compared to the first version in the "To Do List". Let me explain them in detail:

- In the updated iteration of the "To Do List" page, clicking on the "Sign Out" button redirects users to the "Signup" page, whereas in the initial version, selecting the "Sign Out" button would direct them to the "Login" page.
- Rather than utilizing radio buttons to indicate priority levels such as Low, Medium, and High, I've opted for a list format to display the available priority options. It's a different way to select the priority for my tasks.
- Additionally, I've enlarged the image located at the top-center of the page. It now takes up more space on the page.

See these changes in Figure 6 below:

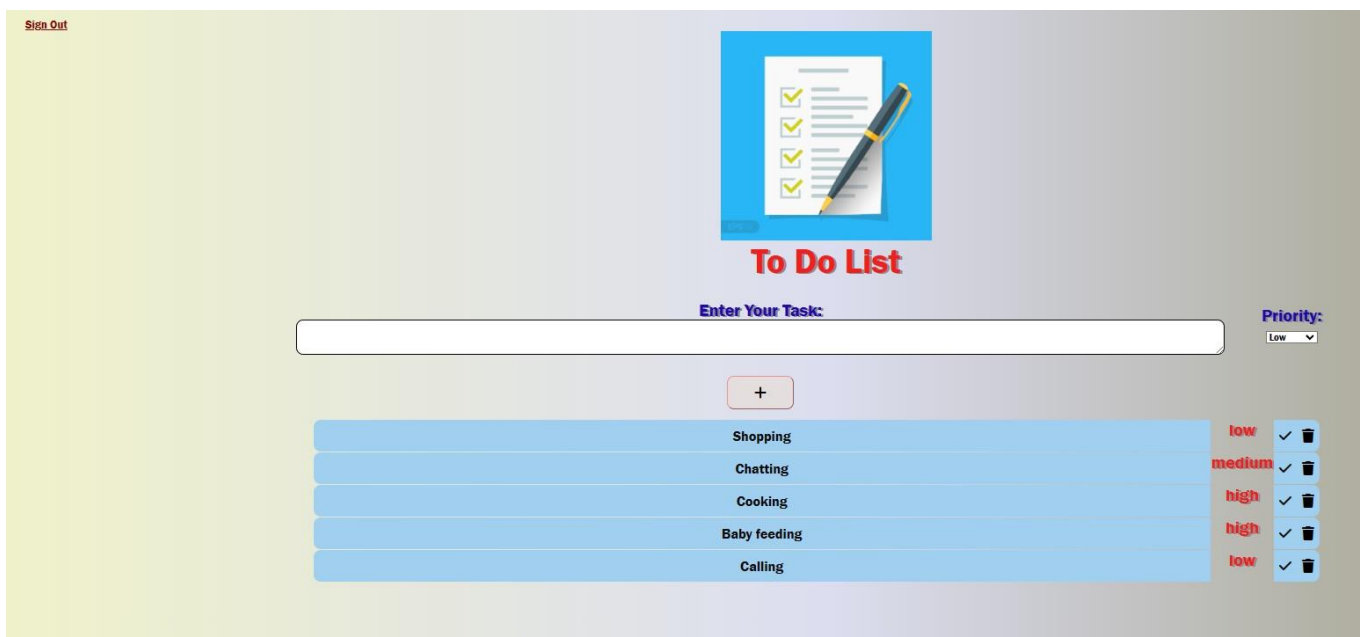


Figure 6: Screenshot of to-do-list page (Version 2)

Description of the GUI Testing Tool:

This GUI testing ensures the functionality of all the GUI elements on different pages. I do this by looking at the things users interact with, like buttons, scrollbars, and menus. There are different ways to do GUI testing, including Automated Testing, Manual Testing, and Model Based Testing. In this project, I am focusing on automated GUI testing.

I did some research to find a good tool for automated testing, and I chose Selenium Integrated Development Environment (Selenium IDE). This tool helps us create test cases automatically. Selenium IDE does the automation in two main ways.

The process of automated GUI testing with Selenium IDE involves two stages: Recoding and Replaying.

- Recording Stage: During the Recording Stage, the automation tool captures and records all the application elements requiring testing. In this phase, users could designate the testing type for each element through a right-click action.
- Replaying Stage: After recording, the test steps are played back or executed on the application being tested. This helps check if the application behaves correctly based on the recorded interactions.

Figure 7 displays the user interface of the Selenium IDE tool, with crucial components for test case creation highlighted using red boxes and arrows. This graphical user interface (GUI) testing application is proficient in generating a range of test cases, which encompass evaluations of GUI element attributes such as location, size, positioning, and functionality within the application.

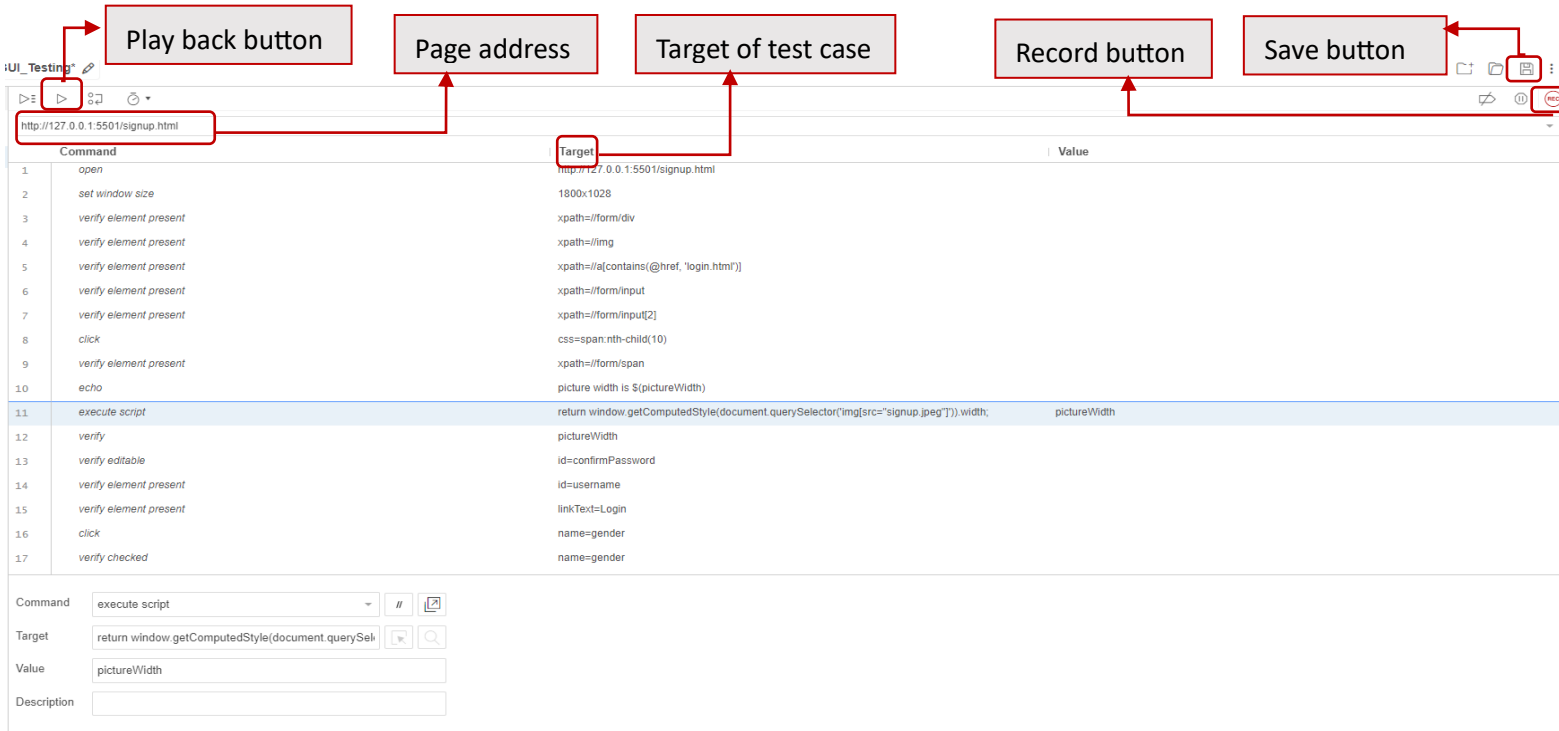


Figure 7: The user interface of Selenium IDE and main components.

General Description of the Test cases and their Coverage:

At first, for the first version, I created test cases by selecting various elements on each page. Then, in the second part, I used the same test cases from the first version. These generally cover aspects like checking if GUI elements exist, their position, size, location, whether they can be edited, how they work, and what type they are. I also tested how the pages connect with each other. I checked if the links between pages exist and work properly. Furthermore, I conducted testing on specific text boxes and buttons by actively interacting with them to ensure their compliance with their intended

functionality. The subsequent section of this report will provide a comprehensive account of the various test cases employed and the outcomes observed for each page.

## Test Case and Results:

Test cases and results of “Signup” page (version 1 and version 2):

Within this section of the report, I will assess the critical test cases for each page and delve into the causes behind test failures in the second iteration of the application. Figures 8a and 8b present the test cases and outcomes for both the initial and revised versions of the “Sign Up” page. Let's proceed to review these specific test cases:

- Test Cases 1 and 2: The following commands are used to access the “Sign Up” page and configure the webpage window size to 1800x1028.
- Test Cases 3 and 4: These test whether the large picture at the top center of the “Sign Up” page exists and is in the correct position.
- Test Case 5: This command verifies whether clicking on the "Login" link successfully redirects the user to the Login page, ensuring seamless page navigation.
- Test Cases 6, 7 and 9: These check positions of the “Submit” and "Reset" buttons, and positions of the "Already have an account?" message.
- Test Case 8: This command confirms the presence of the "Already have an account?" text by simulating a click on it.
- Test Cases 12 and 14: These extract the size of the top center picture on the “Sign Up” page and confirm if its width matches the expected value.
- Test Case 10: The "echo" command prints a message to the console, including the value of the variable picture Width.
- Test Case 15: This verifies if the password field is editable.
- Test Case 16: This command validates the presence of the “Username” field.
- Test Case 17: This checks if the "Login" text functions as a hyperlink.
- Test Cases 18 and 19: These commands are employed to verify whether the radio buttons used for gender selection are checked.
- Test Cases 20 and 21: These extract the location of the "Reset" button on the "Signup" page and confirm if its location matches the expected one.

In the second version of the application, if any of these test cases failed, it means that there are issues with the corresponding elements or functionalities in the new version. Further investigation would be needed to understand why these failures occurred and to make necessary improvements.

Test suites +			
Search tests...			
X Search suites			
Test1_signup_version1	Test2_signup_version2*	Test3_login_version1*	Test4_login_version2
Test5_todolist_version1	Test6_todolist_version2		
Command		Target	Value
1	open	http://127.0.0.1:5501/signup.html	
2	set window size	1800x1028	
3	verify element present	xpath=/form/div	
4	verify element present	xpath=/img	
5	verify element present	xpath=/a[contains(@href, 'login.html')]	
6	verify element present	xpath=/form/input	
7	verify element present	xpath=/form/input[2]	
8	click	css=span:nth-child(10)	
9	verify element present	xpath=/form/span	
10	echo	picture width is \${pictureWidth}	
11	//		
12	execute script	return window.getComputedStyle(document.querySelector('img[src="signup.jpeg"]')).width;	pictureWidth
13	//		
14	verify	pictureWidth	700px
15	verify editable	id=confirmPassword	
16	verify element present	id=username	
17	verify element present	linkText=Login	
18	click	name=gender	
19	verify checked	name=gender	
20	execute script	const rect=document.querySelector('input[type="reset"][name="reset"][value="Reset"]').btn.btn-secondary.getBoundingClientRect();return `\${rect.left},\${rect.top}`;	elementLocation
21	echo	\${elementLocation}	
22	//		
Command open // [img]			
Target http://127.0.0.1:5501/signup.html [img]			

Figure 8 (a): Test cases and results of the “Signup” page (Version 1)

Figure below shows that all the tests have been succeeded in the case of the previous version of then of the Signup page. But there are five failed test cases: numbers 6, 7, 8, 10 and 13, which are highlighted in red. Let's break down why these test cases failed for the current version of Signup page:

- Test Case 6: The test case failed due to the "Login" link redirect to a different page instead of the expected outcome "ToDo List" page rather the expected "Login" page. The flow between the pages was altered in the second version.
- Test Cases 7, 8, and 10: These test cases examine "Submit" and "Reset" button's positions, and the "Already have an account?" text. They failed because the positions of these elements are changed in the second version.
- Test Case 13: This test case assesses image's positioned at the top center of the "Signup" page. It yielded a failure because the image's dimensions are modified in the current version.

These failures indicate that there are significant changes in the layout and functionality of the "Signup" page in the second version. To ensure the new version works correctly, adjustments and corrections should be made based on these test results.

Project: Nigar\_Sultana\_GUI\_Testing\*

Test suites	+	▶▶▶	▶▶▶	▶▶▶	▶▶▶	▶▶▶	▶▶▶
Search tests...	Q	http://127.0.0.1:5500/signup.html					
<div> <div>✗ Search suites</div> <div> <div>✓ Test1_signup_version1</div> <div>✗ Test2_signup_version2*</div> <div>✗ Test3_login_version1*</div> <div>✗ Test4_login_version2</div> <div>✓ Test5_todolist_version1</div> <div>✗ Test6_todolist_version2</div> </div> </div>		Command	Target	Value			
	2	✓ open	http://127.0.0.1:5500/signup.html				
	3	✓ set window size	1800x1028				
	4	✓ verify element present	xpath=/form/div				
	5	✓ verify element present	xpath=/img				
	6	✗ verify element present	xpath=/a[contains(@href, 'login.html')]				
	7	✗ verify element present	xpath=/form/input				
	8	✗ verify element present	xpath=/form/input[2]				
	9	//					
	10	✗ verify element present	xpath=/form/span				
	11	✓ echo	picture width is \$(pictureWidth)				
	12	✓ execute script	return window.getComputedStyle(document.querySelector('img[src="signup.jpeg"]')).width;	pictureWidth			
	13	✗ verify	pictureWidth	700px			
	14	✓ verify editable	id=confirmPassword				
	15	✓ verify element present	id=username				
	16	✓ verify element present	linkText=Login				
	17	✓ click	name=gender				
	18	✓ verify checked	name=gender				
	19	✓ execute script	const rect=document.querySelector("input[type='reset']")[name='reset']?btn.btn-se	elementLocation			
	20	✓ echo	\$(elementLocation)				
	21	✓ close					
		Command	open	//			
		Target	http://127.0.0.1:5500/signup.html				

Figure 8(b): Test cases and results of the “Signup” page (version 2)

Test cases and results of Login page (version 1 and version 2):

In Figures 9(a) and 9(b), I can see all the test cases and the result for the previous and the current version of the Login page. Here's an explanation of the test cases for the Login page:

- Test Cases 1 and 2: These commands initiate the opening of the Login page and configure the window size.
- Test Case 3: This test case validates the presence of the "Sign Up" text, which links the Login page to the Signup page.
- Test Case 4: It confirms the positioning of the large image on Login page’s right side.
- Test Case 6: This test case ensures that clicking the "Sign Up" button successfully redirects to the Signup page, maintaining a smooth transition between these two pages.
- Test Case 7: It checks for the existence of "Submit" element.
- Test Cases 9 and 11: These test cases assess Username" text field’s existence and editability.
- Test Case 10: This ensures the placement of the key picture on Login text’s left side.
- Test Cases 12 and 13: These test cases evaluate Password text field’s presence and the editability.
- Test Cases 14 through 16: These commands obtain the size of the key image situated on the left side of the Login text and subsequently ascertain whether the extracted width corresponds to the specified desired value.
- Test Cases 16 to 19: These instructions retrieve the precise coordinates of the "Submit" button's location on the Login page and subsequently validate whether the extracted location corresponds to the desired location.

For the first version of the Login page, all test cases passed. However, in Figure 9(b), three test cases failed in the second version:

- Test Case 4: The failure occurred because the position of the large image was altered in the second version of the application.
- Test Case 6: In the second version, the test case failed due to the "Sign Up" link redirecting the user to the "To Do List" page instead of the expected "Signup" page.
- Test Case 16: Changes in the main image’s size located on the Login text’s left side caused this test case to fail.



These failures indicate that the current version of the Login page has undergone significant changes, affecting the positions, flow between pages, and image sizes. Further adjustments and corrections are needed to align it with the expected behavior.

Project: Nigar\_Sultana\_GUI\_Testing

Executing ▾	⏮ ⏪ ⏩ ⏭ ⌛ ▾		
✓ Test1_signup_version1	http://127.0.0.1:5500/todolist.html		
X Test2_signup_version2			
✓ Test3_login_version1			
X Test4_login_version2			
✓ Test5_todolist_version1			
X Test6_todolist_version2			
	Command	Target	Value
1	✓ open	http://127.0.0.1:5501/login.html	
2	✓ set window size	1800x1028	
3	✓ verify element present	linkText=Sign Up	
4	✓ verify element present	xpath=//div[2]/img	
5	// ✓		
6	✓ verify element present	xpath=//a[contains(@href, 'signup.html')]	
7	✓ verify element present	name=submit	
8	// ✓		
9	✓ click	id=username	
10	✓ verify element present	xpath=//img	
11	✓ assert editable	id=username	
12	✓ click	id=password	
13	✓ assert editable	id=password	
14	✓ execute script	return window.getComputedStyle(document.querySelector('img[src="login.png"]')).width;	extractedWidth
15	✓ echo	image width is \${extractedWidth}	
16	✓ verify	extractedWidth	40px
17	✓ execute script	const rect=document.querySelector('input[type="submit"][name="submit"][value="Submit"].btn.btn-primary').getBoundingClientRect();return `\${rect.left},\${rect.top}`;	elementLocation
18	✓ echo	\${elementLocation}	
Command open // 📄			
Target http://127.0.0.1:5501/login.html 🗑 🔍			

Figure 9(a): Test cases and results of “Login” page (version 1)

Project: Nigar\_Sultana\_GUI\_Testing

Executing ▾	⏮ ⏪ ⏩ ⏭ ⌛ ▾		
✓ Test1_signup_version1	http://127.0.0.1:5500/todolist.html		
X Test2_signup_version2			
✓ Test3_login_version1			
X Test4_login_version2			
✓ Test5_todolist_version1			
X Test6_todolist_version2			
	Command	Target	Value
1	✓ open	http://127.0.0.1:5500/login.html	
2	✓ set window size	1800x1028	
3	✓ verify element present	linkText=Sign Up	
4	X verify element present	xpath=//div[2]/img	
5	✓ verify element present	name=submit	
6	X verify element present	xpath=//a[contains(@href, 'signup.html')]	
7	✓ click	css= form-group:nth-child(1)	
8	✓ assert element present	css= form-group:nth-child(1)	
9	✓ verify element present	xpath=//img	
10	✓ click	id=username	
11	✓ assert editable	id=username	
12	✓ click	id=password	
13	✓ assert editable	id=password	
14	✓ execute script	return window.getComputedStyle(document.querySelector('img[src="login.png"]')).width;	extractedWidth
15	✓ echo	image width is \${extractedWidth}	
16	X verify	extractedWidth	40px
17	✓ execute script	const rect=document.querySelector('input[type="submit"][name="submit"][value="Submit"].btn.btn-primary').getBoundingClientRect();return `\${rect.left},\${rect.top}`;	elementLocation
18	✓ echo	\${elementLocation}	
19	✓ close		
Command open // 📄			
Target http://127.0.0.1:5500/login.html 🗑 🔍			

Figure 9(b):Test cases and results of “Login” page (version 2)

Test cases and results of “To Do List” page (version 1 and version 2):

In Figures 10(a) and 10(b), I can examine the test cases and their corresponding results for both the initial and subsequent versions of the “To Do List” page. Here's a breakdown for the “To Do List” page’s test cases:

- Test Cases 1 and 2: These commands open the page and set the dimension of the window.
- Test Case 3: This test case identifies the priority level of the page.
- Test Case 4: It checks the image position at the top of the page.
- Test Case 6: This test case checks the radio button on the “To Do List” page.
- Test Cases 7 and 8: These tests identify the element type which are used to select priority level.
- Test Case 9: This test case verifies the presence of the text field designated for entering tasks.
- Test Cases 10 and 11: These identify the existence of the list text at the top of the page.
- Test Case 12: It checks the existence of priority level’s medium level.
- Test Cases 13 to 15: These involve extracting the image size at the top middle of the “To Do List” page to make sure the width of the image matches with the expected value.
- Test Case 16: This confirms that clicking the Sign Out hyperlink redirects to the “Login page”, to make sure the navigation between the “To Do List” and “Login” pages.
- Test Cases 17 and 18: These extract the exact Plus button’s location used to incorporate tasks in the page and verify the location of the tasks matches the expected.
- 

For the “To Do List” page’s version one, all tests passed. However, in Figure 10(b), six test cases failed in the second version:

- Test Case 6: The failure occurred because the position of the "High" priority button was altered in the second version of the application.
- Test Cases 7 and 8: These tests failed due to the radio button which is used to select the important levels and Ire replaced with a list in the second version.
- Test Case 12: It failed because the Medium priority label was not included in version two.
- Test Case 15: This test case resulted in a failure due to a modification in the size of the picture located at the top center of the page.
- Test Case 16: Sign Out button in version two of the page redirects to the “Sign Up” page in lieu of the “login” page, which causes the test to fail.

These failures indicate that the second version of the "To Do List" page went under significant changes, including the priority selection method, label changes, image dimensions, and page flow alterations. Adjustments are needed to align it with the expected behavior.

Project: Nigar_Sultana_GUI_Testing			
Executing ▾	<div> <div>⏮</div> <div>⏪</div> <div>⏩</div> <div>⏭</div> <div>⌂</div> <div>⌵</div> </div>		
<div> <div>✓ Test1_signup_version1</div> <div>✗ Test2_signup_version2</div> <div>✓ Test3_login_version1</div> <div>✗ Test4_login_version2</div> <div>✓ Test5_todolist_version1</div> <div>✗ Test6_todolist_version2</div> </div>	http://127.0.0.1:5500/todolist.html		
	Command	Target	Value
	1 ✓ open	http://127.0.0.1:5501/todolist.html	
	2 ✓ set window size	1800x1028	
	3 ✓ verify element present	css= priority-container>label:nth-child(1)	
	4 ✓ verify element present	xpath=//img	
	5 ✓ verify element present	xpath=//button[@type='button']	
	6 ✓ verify element present	xpath=//label[4]	
	7 ✓ verify element present	xpath=//input[@id='low']	
	8 ✓ verify element present	xpath=//input[@id='high']	
	9 ✓ verify element present	name=text	
	10 ✓ click	xpath=//h1	
	11 ✓ verify element present	xpath=//h1	
	12 ✓ verify element present	xpath=//label[contains(.,'Medium')]	
	13 ✓ execute script	return window.getComputedStyle(document.querySelector('img[src="todo_list.png"]')).width;	extractedWidth
	14 ✓ echo	Image width is \${extractedWidth}	
	15 ✓ verify	extractedWidth	200px
	16 ✓ verify element present	xpath=//a[contains(@href, 'login.html')]	
	17 ✓ execute script	const rect=document.querySelector("button[type='button']").buttoninput i fa-solid fa-plus).getBoundingClientRect();return `\${rect.left},\${rect.top}`;	elementLocation
	18 ✓ echo	\${elementLocation}	
	19 ✓ close		
<div> <div>Command</div> <div>open</div> <div>⏮</div> <div>⏪</div> <div>⏩</div> <div>⏭</div> <div>⌂</div> <div>⌵</div> </div>			
<div> <div>Target</div> <div>http://127.0.0.1:5501/todolist.html</div> <div>⏮</div> <div>⏪</div> <div>⏩</div> <div>⏭</div> <div>⌂</div> <div>⌵</div> </div>			

Figure 10(a): Test cases and results of “To Do-List” page (version1)

Project: Nigar_Sultana_GUI_Testing			
Executing ▾	<div> <div>⏮</div> <div>⏪</div> <div>⏩</div> <div>⏭</div> <div>⌂</div> <div>⌵</div> </div>		
<div> <div>✓ Test1_signup_version1</div> <div>✗ Test2_signup_version2</div> <div>✓ Test3_login_version1</div> <div>✗ Test4_login_version2</div> <div>✓ Test5_todolist_version1</div> <div>✗ Test6_todolist_version2</div> </div>	http://127.0.0.1:5500/todolist.html		
	Command	Target	Value
	1 ✓ open	http://127.0.0.1:5500/todolist.html	
	2 ✓ set window size	1800x1028	
	3 ✓ verify element present	css= priority-container>label:nth-child(1)	
	4 ✓ verify element present	xpath=//img	
	5 ✓ verify element present	xpath=//button[@type='button']	
	6 ✗ verify element present	xpath=//label[4]	
	7 ✗ verify element present	xpath=//input[@id='low']	
	8 ✗ verify element present	xpath=//input[@id='high']	
	9 ✓ verify element present	name=text	
	10 ✓ click	xpath=//h1	
	11 ✓ verify element present	xpath=//h1	
	12 ✗ verify element present	xpath=//label[contains(.,'Medium')]	
	13 ✓ execute script	return window.getComputedStyle(document.querySelector('img[src="todo_list.png"]')).width;	extractedWidth
	14 ✓ echo	Image width is \${extractedWidth}	
	15 ✗ verify	extractedWidth	200px
	16 ✗ verify element present	xpath=//a[contains(@href, 'login.html')]	
	17 ✓ execute script	const rect=document.querySelector("button[type='button']").buttoninput i fa-solid fa-plus).getBoundingClientRect();return `\${rect.left},\${rect.top}`;	elementLocation
	18 ✓ echo	\${elementLocation}	
	19 ✓ close		
<div> <div>Command</div> <div>open</div> <div>⏮</div> <div>⏪</div> <div>⏩</div> <div>⏭</div> <div>⌂</div> <div>⌵</div> </div>			
<div> <div>Target</div> <div>http://127.0.0.1:5500/todolist.html</div> <div>⏮</div> <div>⏪</div> <div>⏩</div> <div>⏭</div> <div>⌂</div> <div>⌵</div> </div>			

Figure 10(b): Test cases and results of “To-Do-List” page (version 2)

### Assessment of the tool:

Selenium IDE is a robust test automation tool designed for functional testing. In this project, I evaluated its usability, coverage capabilities, and tool features which are described below.

Usability:

- **Learnability:** Installing Selenium IDE as a browser extension was straightforward. Its simple user interface and intuitive recording feature make it accessible even to users without extensive coding experience.
- **Memorability:** Once I become familiar with Selenium IDE, it's easy to remember how to create test cases and apply the acquired knowledge to similar scenarios. The tool's consistent design contributes to its memorability.
- **Generating Errors:** Selenium IDE provides detailed error messages through its Log, which makes it easy to realize and resolve errors. This enhances the tool's efficiency and productivity.
- **Efficiency:** Selenium IDE automates repetitive tasks, allowing for fast test execution and quick adjustments if needed. It also allows to control the execution speed of the test.
- **Subjective Satisfaction:** Overall, the tool offers robust functionality, user-friendly interface, effective error management, and an easy learning environment.

#### Coverage Provided:

- Selenium IDE allow me to record all the elements necessary to test during the recording phase.
- It allows the flexibility of specific test type to elements easily by right-clicking on them.
- The tool supported the generation of a wide range of test cases, enabling verification of various functionalities, including data input, radio buttons, checkboxes, lists, and GUI element presence.
- Selenium IDE facilitated the generation of automotive and semi-automotive test cases to assess the position, size, and precise location of GUI elements.
- I successfully tested the application on different pages and the flow from one page to another by creating test cases to verify reference links.

In summary, Selenium IDE effectively met the project's objectives and performed well in terms of test coverage, allowing me to assess various aspects of the application's functionality.

#### Tool Features:

- Selenium IDE's user interface includes record and playback buttons, simplifying the testing process.
- During recording, I could right-click on elements to assign specific types of testing.
- The automation tool automatically generated test cases for various elements, each comprising three distinct sections: Command, Target, and Value.
- I could modify the objective of a test case by clicking on the Target box.
- Selenium IDE offers a range of Command options, allowing me to perform various types of testing during recording.
- The tool provides flexibility in testing element positions using options like "xpath:position."
- Selenium IDE allows me to verify actions such as hyperlink redirection, as shown in Figure 12b.

In conclusion, Selenium IDE's features simplify the process of generating and executing test cases, making it a valuable tool for functional testing of GUI applications.

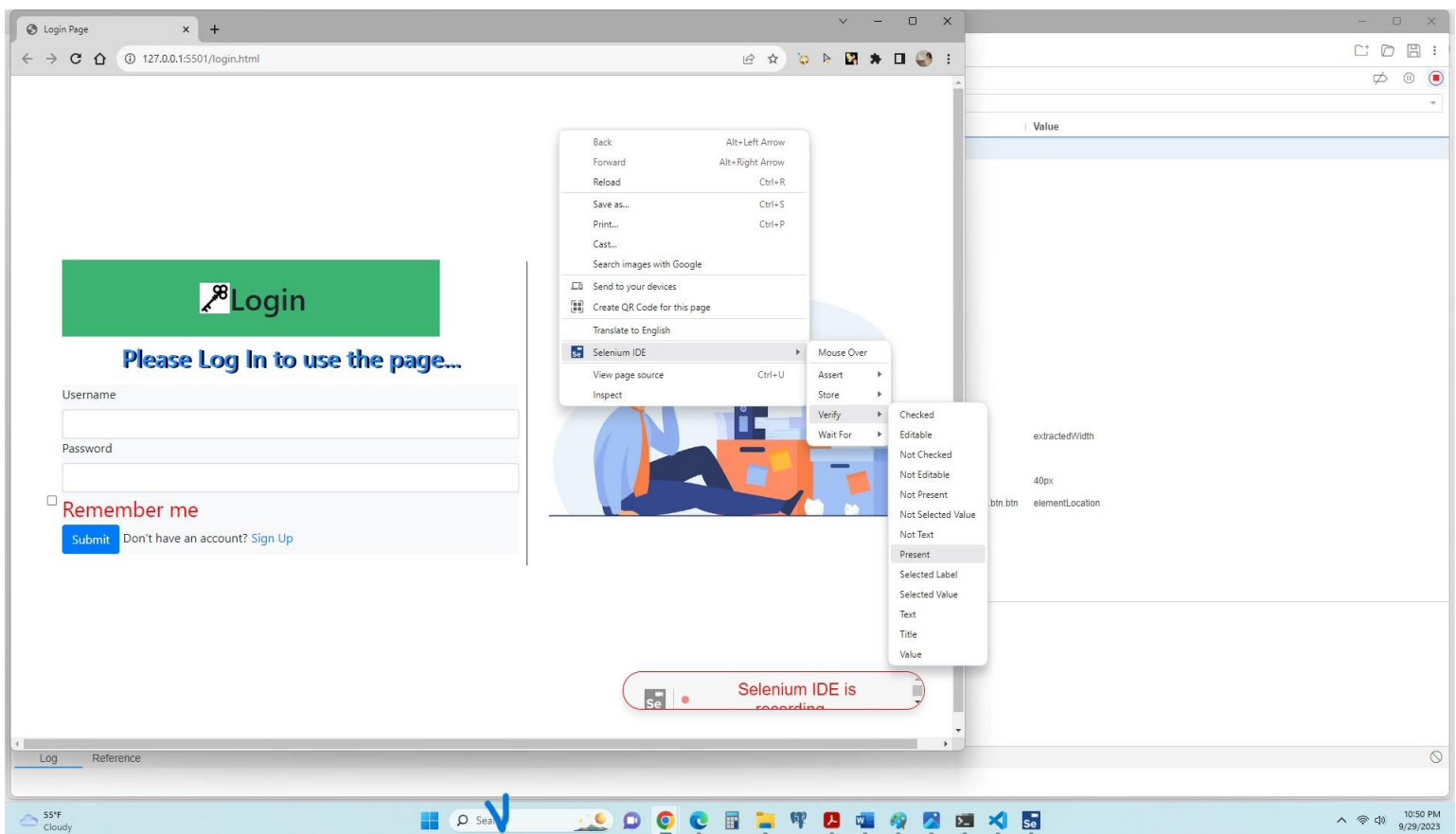


Figure :11 Different types of test categories

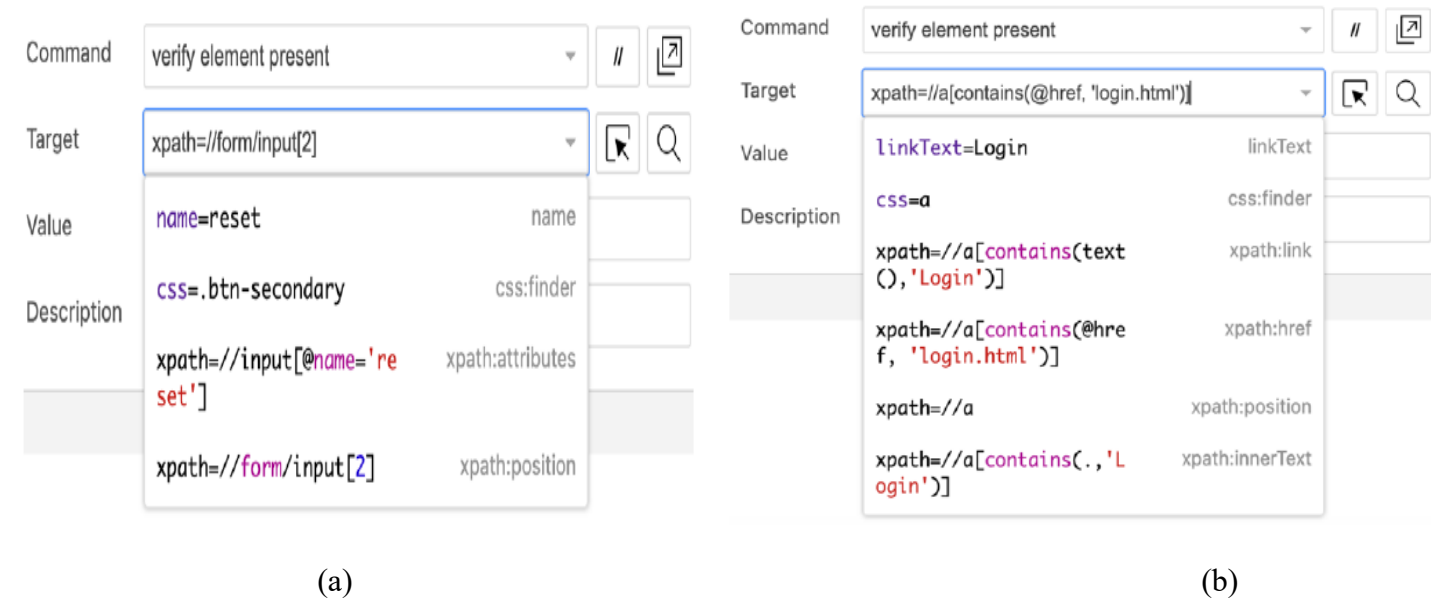


Figure:12. Screenshots of different Target options: (a) position of Reset button, and (b) reference link to “Login” Page

### Evaluation of the reusability of test cases and results:

The reusability of test cases in this project was a significant goal, and Selenium IDE proved to be effective in achieving this objective. Here's an evaluation of the reusability of test cases and the results:

- **Duplicating Test Cases:** Initially, a set of test cases was developed for the first version of the application. These test cases were then duplicated for testing the second version. Selenium IDE allowed for easy duplication of existing test cases, saving time and effort.
- **Adapting the "Open" Command:** For assessing the second iteration of the application, the sole adjustment required involved substituting the "Open" command to access the second version in place of the initial one. This uncomplicated alteration facilitated the utilization of identical test cases for both versions.

- **Handling Changes:** As per project specifications, changes are made to the second version of the application. Test results revealed that the test cases targeting elements subjected to changes did indeed fail. This demonstrates that Selenium IDE successfully detected alterations in size, position, and location of elements.
- **Detection of Flow Changes:** Selenium IDE also detected changes in the flow between different pages of the application. For example, when clicking on elements that should redirect to different pages, Selenium IDE accurately identified if the redirections are as expected.
- **Reliability of Selenium IDE:** The outcomes of this project indicate that Selenium IDE is a convenient testing tool for GUI elements. All test cases targeting unchanged elements passed, highlighting its effectiveness in verifying stable parts of the application.

Selenium IDE facilitated the reusability of test cases and effectively detected changes between the first and second versions of the application. It proved to be a dependable tool for GUI testing, ensuring that both modified and unchanged elements are tested and evaluated accurately.

## References:

- a. Qi, S. "Evaluation of the Maintenance Required by Ib Application Test Suites." Doctoral dissertation, Politecnico di Torino, 2020.
- b. Vaswani, G. A. "A Blockchain-Based Approach for Securing Electronic Hospital Records." Doctoral dissertation, National College of Ireland, 2022.
- c. "Selenium IDE." SeleniumHQ, <https://www.seleniumhq.org/selenium-ide/>.
- d. "QAfox Selenium Automation Tutorial." QAfox, <https://www.qafox.com/selenium/selenium-automation-tutorial/>.
- e. "Introduction to Selenium IDE - A Comprehensive Guide." Testsigma, <https://testsigma.com/blog/selenium-ide/>.