Project #3 Graphical User Interface Testing Project

Niloufar Roozegar

School of Computing and Augmented Intelligence, Arizona State University

CSE 565: Software Verification and Validation

Dr. Ayca Tuzmen

July 2023

Description of the Application

I have developed a Graphical User Interface (GUI) web application consists of three pages: a "Signup" page, a "Login" page, and "To Do List" page. Each page contains various required features, such as images, text boxes, lists, scrollbars, checkboxes, buttons, and labels, which I will describe in more detail for each respective page.

First Version of Application

The "Signup" page incorporates features such as an image and multiple text boxes for inputting personal information such as name, email, password, and phone number. Additionally, I have designed a radio button to enable the selection of gender (male or female), as well as submit and reset buttons (refer to Figure 1). To ensure a smooth user experience, a link has been established between the "Signup" page and the "Login" page for individuals who already possess an account.

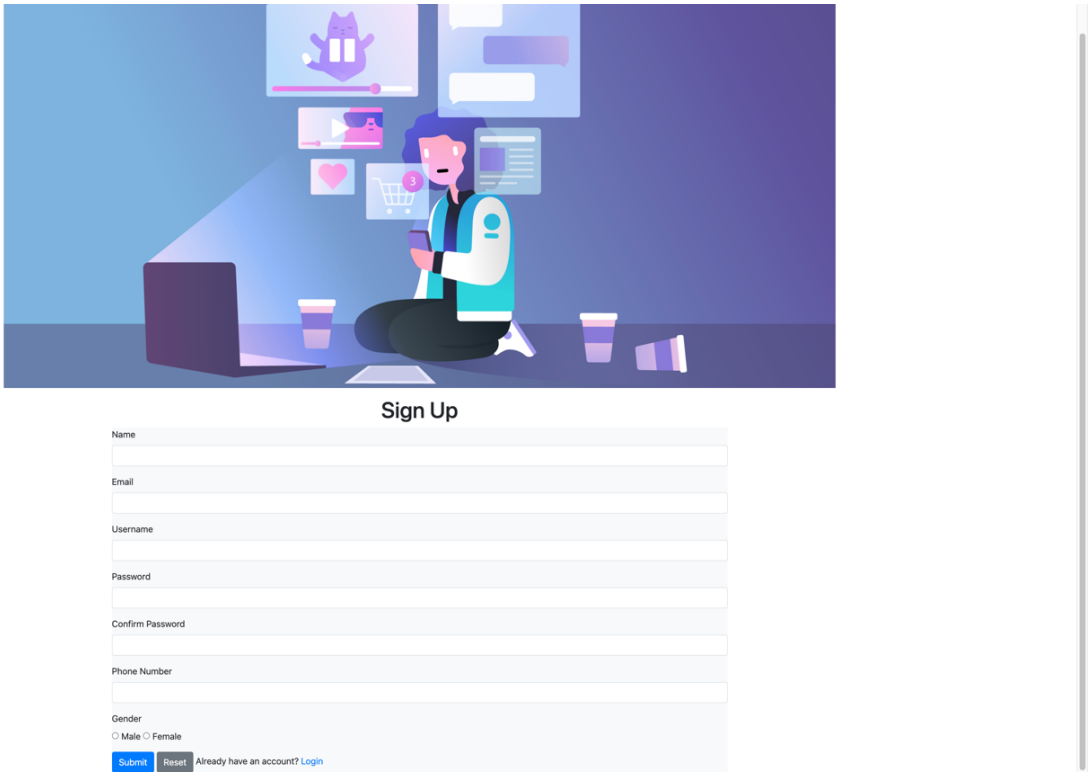


Figure 1. Screenshot of the first version of signup page

As can be seen in Figure 2, the “Login” page comprises an image, two text boxes for entering a username and password, and a “Remember me” checkbox. To facilitate information submission and storage, a submit button has been included. Similar to the previous page, this page is also linked to the “Signup” page to accommodate users without an existing account.

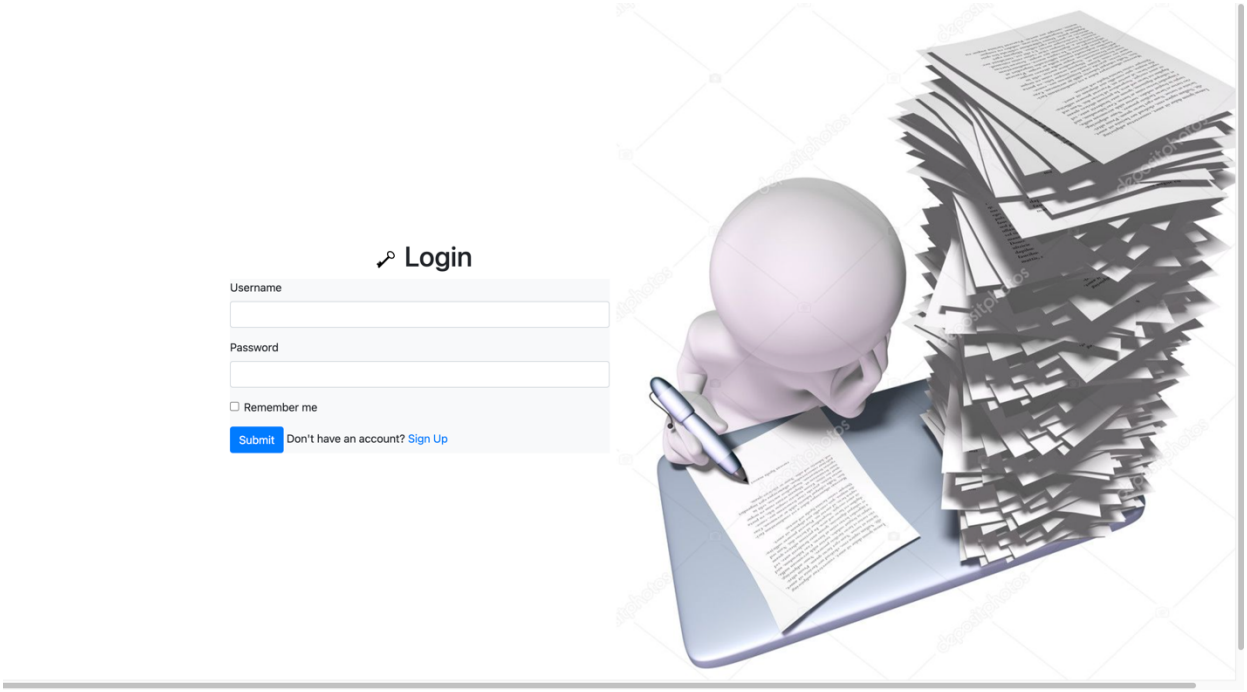


Figure 2. Screenshot of the first version of login page

Lastly, the “To Do List” page features an image and a text box for entering tasks. Furthermore, I have implemented a priority radio button feature that allows users to indicate the urgency of a task by selecting from options such as Low, Medium, and High. To ensure proper task management, each task is accompanied by a corresponding radio button (see Figure 3). The “Sign Out” button is located in the top-left corner of this page, and it is linked to the “Login” page. This allows users to sign out of their current account and login with a different account, providing them with the necessary flexibility and control.

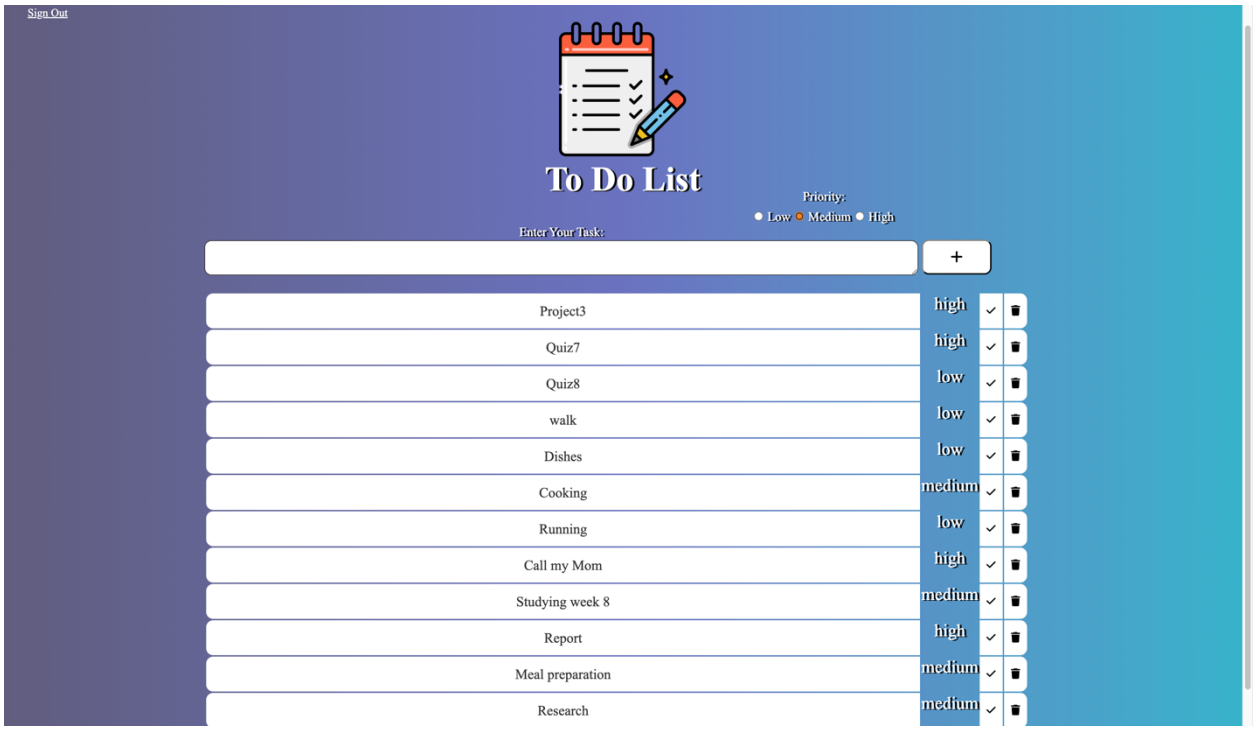


Figure 3. Screenshot of the first version of to-do list page

Second Version of Application

For the second version of my GUI application, I have made several changes to each page, including modifications to the orientation, size, and location of text boxes, images, buttons, etc. As required, I have also altered the flow between the pages. In the following paragraphs, I will present an overview of the modifications made to each page, providing a detailed description of the implemented changes.

As illustrated in Figure 4, some modifications were made to the second version of the “Signup” page. The changes include relocating the two buttons (Submit, Reset), adjusting the placement of login link, and altering the flow of login link situated in the middle bottom of the page.

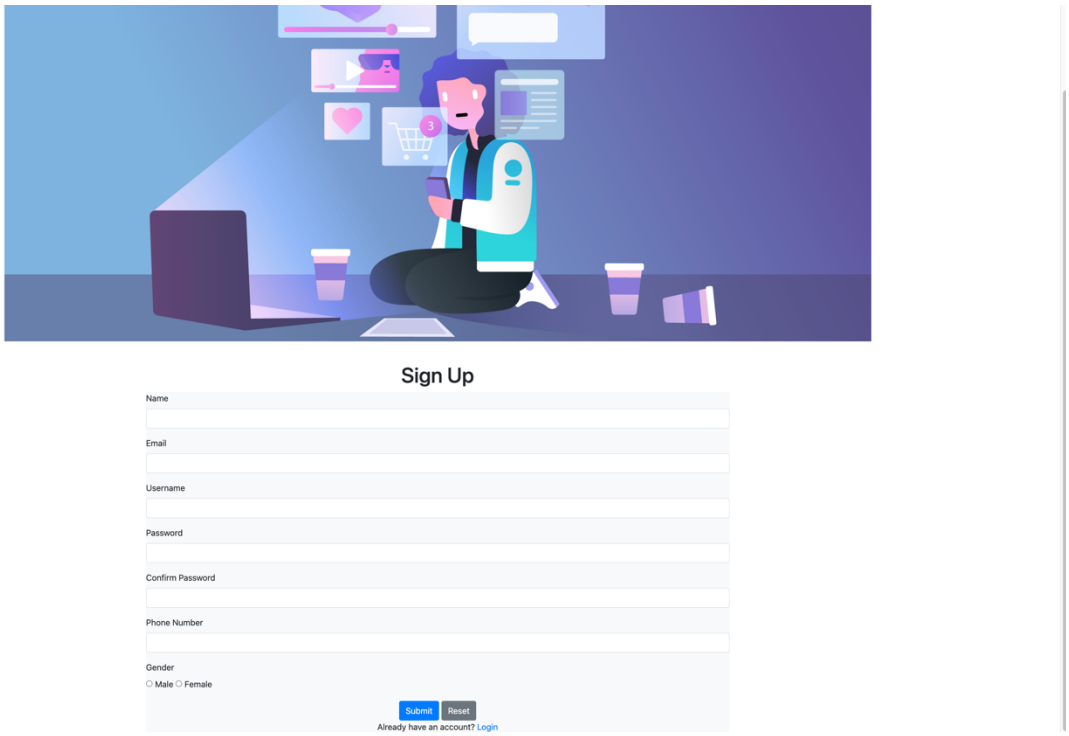


Figure 4. Screenshot of the second version of signup page

Furthermore, changes have been applied to the “Login” page as well. As can be seen in Figure 5, the image on this page has been relocated, swapping its position with the login forms (e.g., username, password, etc.). The size of the key image located on the left side of the “Login” text has been increased. Additionally, to modify the flow between the login and signup pages, clicking on the “Sign Up” button directs the user to the “To Do List” page.

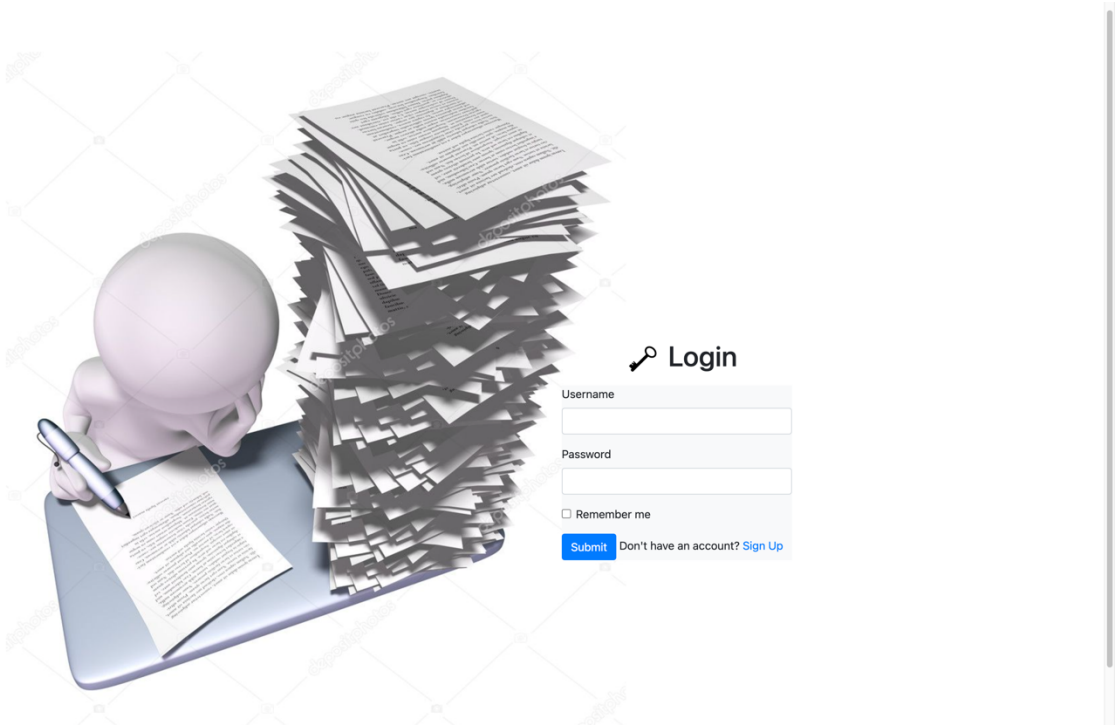


Figure 5. Screenshot of the second version of login page

The second version of the “To Do List” page has been generated by implementing three major changes compared to the first version. Firstly, the size of the picture located at the top center of the page has been increased. Secondly, the radio buttons representing different priority levels (i.e., Low, Medium, and High) have been replaced with a list that displays the priority levels. Lastly, in this new version of the “To Do List” page, clicking on the “Sign Out” button directs the user to the “Signup” page. However, in the first version, the “Sign Out” button was linked to the “Login” page. Figure 6 illustrates the second version of the “To Do List” page.

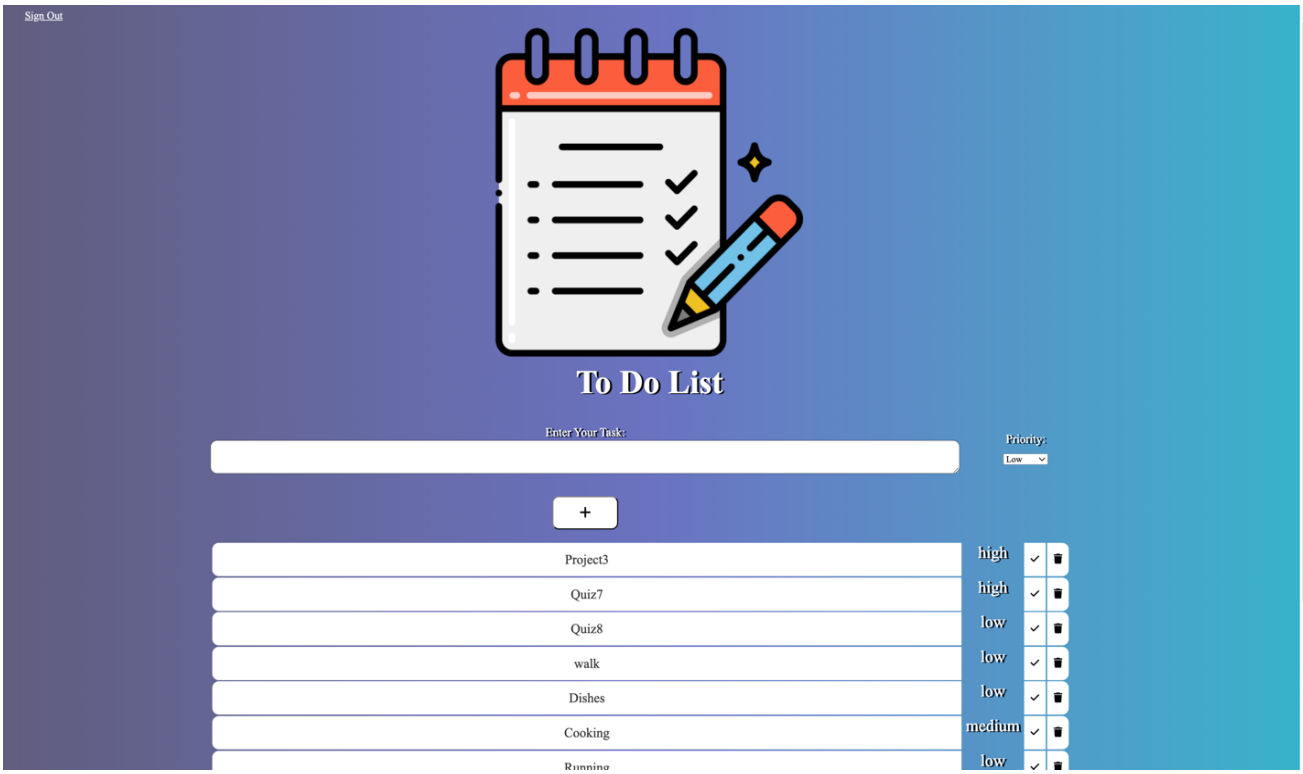


Figure 6. Screenshot of the second version of to-do list page

Description of the GUI Testing Tool

The main objective of Graphical User Interface (GUI) testing is to assess the functionality of an application by examining the various elements that the user interacts with, such as buttons, scrollbars, and drop-down menus. There are different GUI testing techniques such as Manual Based Testing, Model Based Testing, and Automated Testing. The focus of this project is on automated GUI testing. After conducting research regarding the features and usability of various GUI testing tools that generate automated test cases, Selenium's Integrated Development Environment (i.e., Selenium IDE) has been selected for generating the test cases. Selenium IDE GUI testing tool performs the automation in two stages: Recoding and Replaying (Qi, 2020; Vaswani, 2022). Firstly, in the recording phase, the automation tool can capture all the elements that are required to be tested. In this phase, the user can assign a specific type of testing by right-clicking on the desired element. Then, during the playback

phase, the recorded test steps are executed on the application under test. Figure 7 illustrates the user interface of Selenium IDE tool, where the main components of the tool required during test case generation are identified using red boxes and arrows. This GUI testing application can generate various types of test cases, including but not limited to testing the position, size, location, and functionality of the elements.

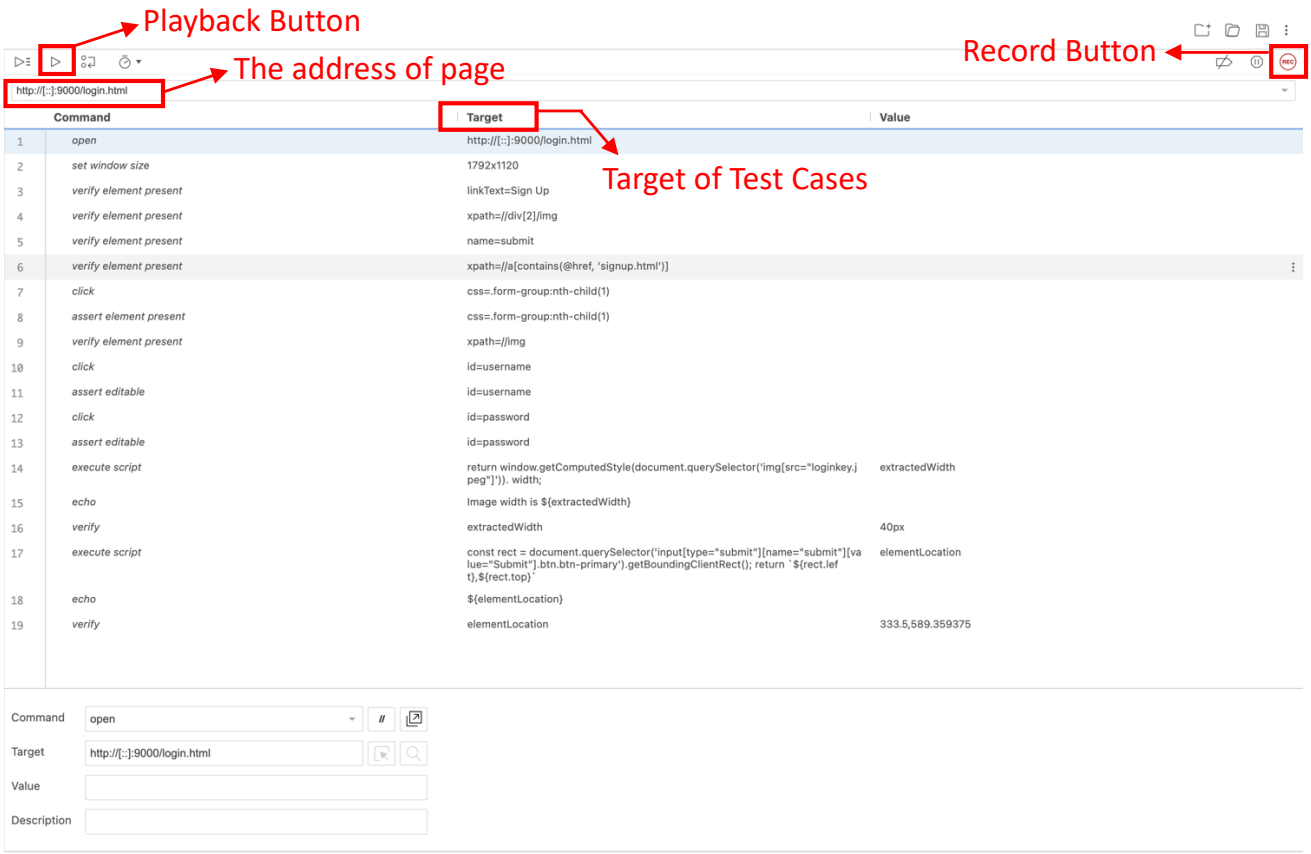


Figure 7. The user interface of Selenium IDE and main components required during test case generation

General Description of the Test Cases and Their Coverage

Initially, test cases were developed by selecting various GUI elements from each page of the first version of the application. Then, the same test cases were used to examine the second version of the application. In general, the test cases cover examining the existence, position, size, location, editability, functionality, and type of GUI elements. The flow between different pages of the application was tested

by selecting the element that links the pages, where the existence of the link between each of the two pages was tested. In addition, the functionality of some text boxes and buttons were tested by clicking on the target GUI element. The following section of this report presents a detailed description of different test cases and the testing results obtained for each page.

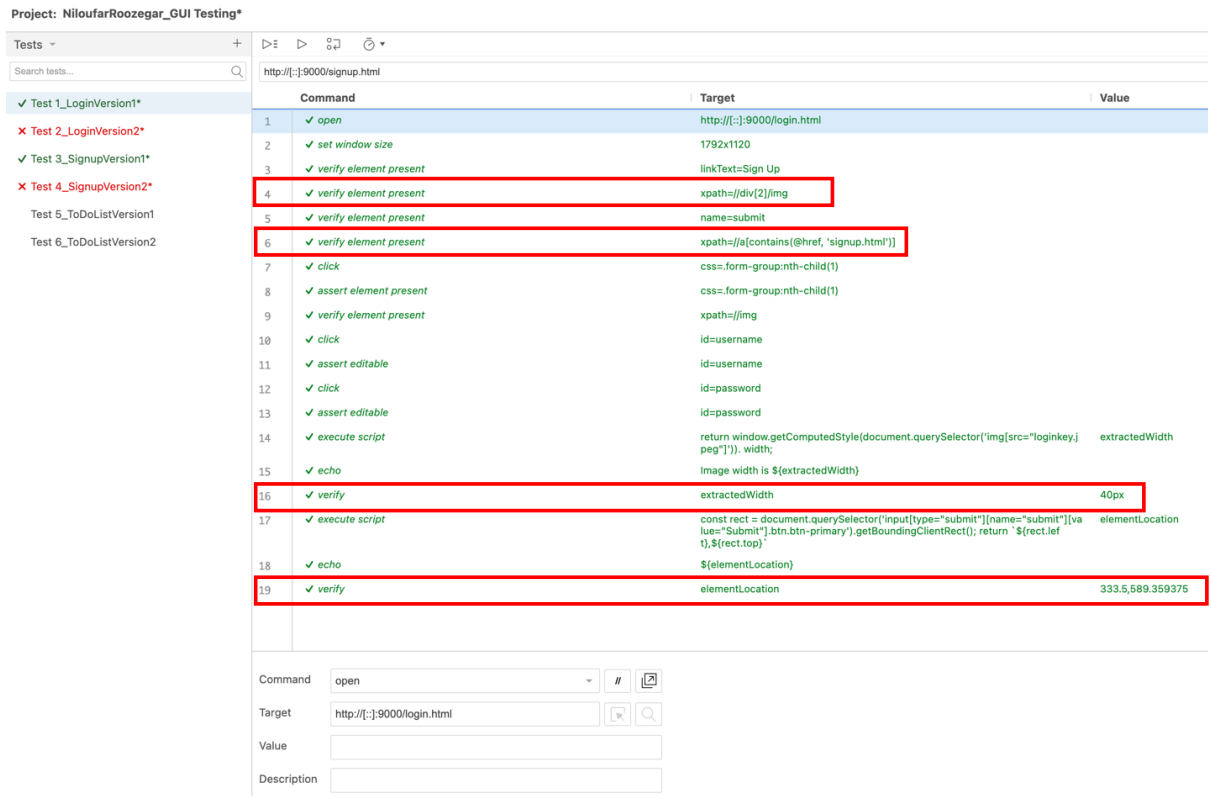
Test Cases and Results

In this section of the report the most important test cases generated for each page will be evaluated and the reasons of test failures in the second version of the application will be discussed. Figures 8a and 8b illustrate the test cases and testing results of the first and second version of the "Login" page. As can be seen, the first two commands open the "Login" page and set the webpage window to the size of 1792×1120. The following bullet points elaborate on the remaining test cases generated for the "Login" page:

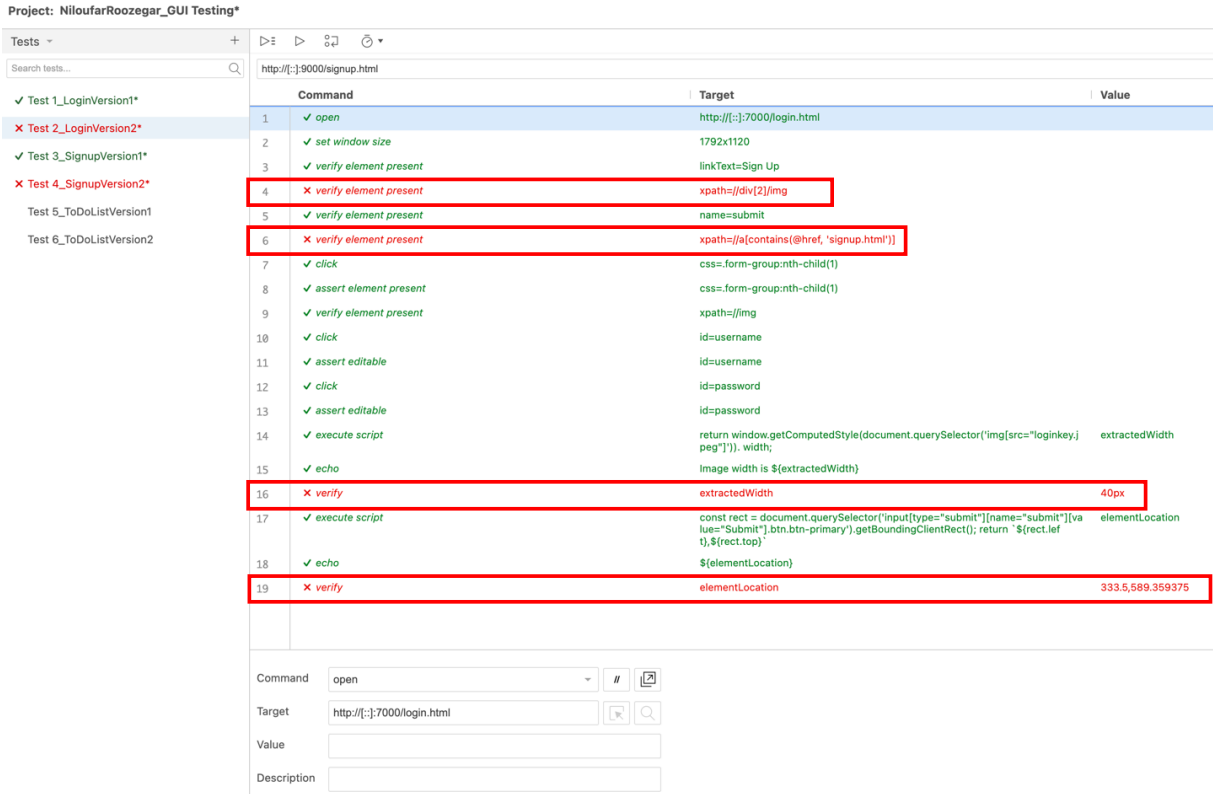
- Test case number three (3) verifies the existence of "Sign Up" text that links the "Login" page to the "Signup" page.
- Test case number four (4) verifies the position of the big image located on the right side of the "Login" page.
- Test case number five (5) verifies the presence of an element named "Submit".
- Test case number six (6) confirms that the "Sign Up" button will redirect the user to the "Signup" page. In other word, it checks whether there is a flow between "Login" and "Signup" pages of the application.
- Test cases number seven (7) and eight (8) evaluate the presence of the form used to enter the "Username" and "Password".
- Test case number nine (9) verifies the position of the key image located on the left side of the "Login" text.

- Test cases number ten (10) through thirteen (13) evaluate the existence and editability of "Username" and "Password" text fields.
- Test cases number fourteen (14) through sixteen (16) extract the size of key image located on the left side of the "Login" text and verifies if the extracted width of the image is the same as the desired value.
- Test cases number sixteen (16) to nineteen (19) extract the exact location (in terms of coordinates) of "Submit" button on the "Login" page and verifies if the extracted location is the same as the desired location.

As can be seen, all the test cases passed for the first version of the "Login" page. According to Figure 8b, a total of four (4) test cases, test cases four (4), six (6), sixteen (16), and nineteen (19), failed upon running the test cases on the second version of the "Login" page. The failed test cases are identified using a red box. Test case number four (4) failed since the position of the big picture changed in the second version of the "Login" page. In this version of the application, clicking on the "Sign Up" link redirects the user to the "To Do List" page instead of the "Signup" page. Therefore, test case number six (6), which verifies the flow between the "Login" and "Signup" pages, failed. Changing the size of the key image located on the left side of the "Login" text was resulted in failure of test case number sixteen (16). Finally, test case number nineteen (19) failed since the location of the "Submit" button changed in the second version of the application.



(a)



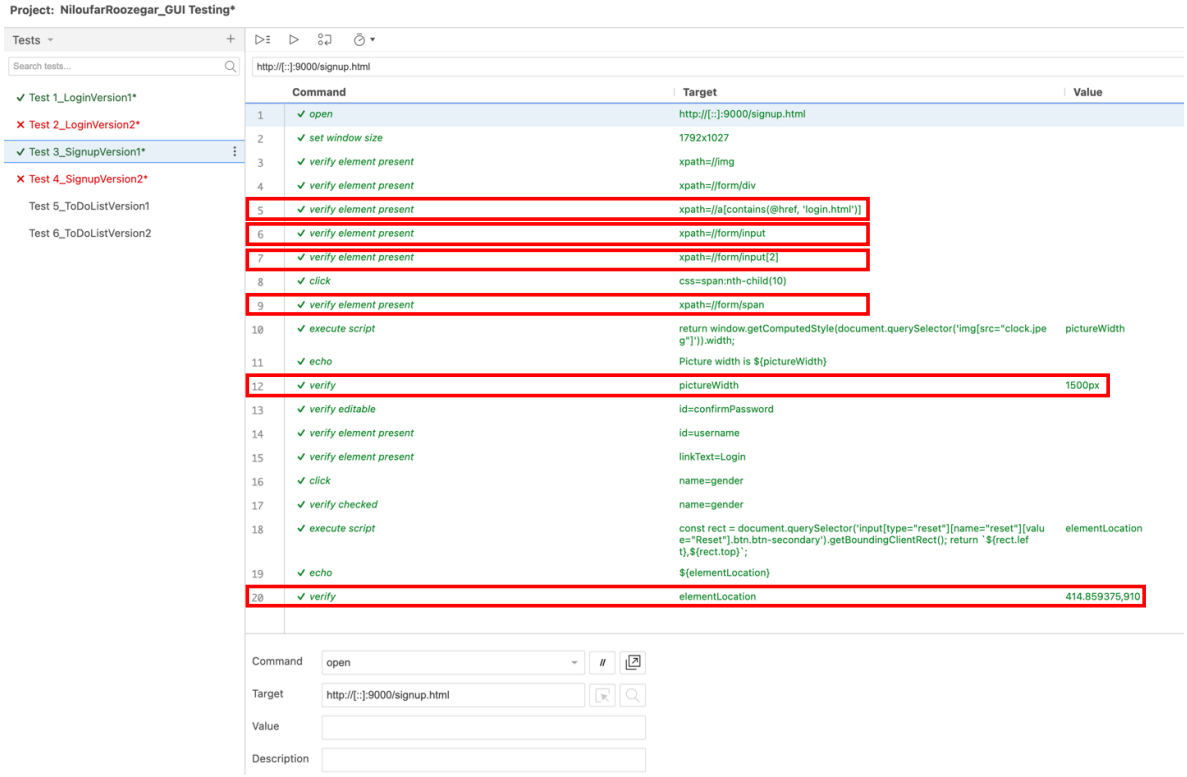
(b)

Figure 8. The test cases and results of the (a) first and (b) second version of the “Login” page

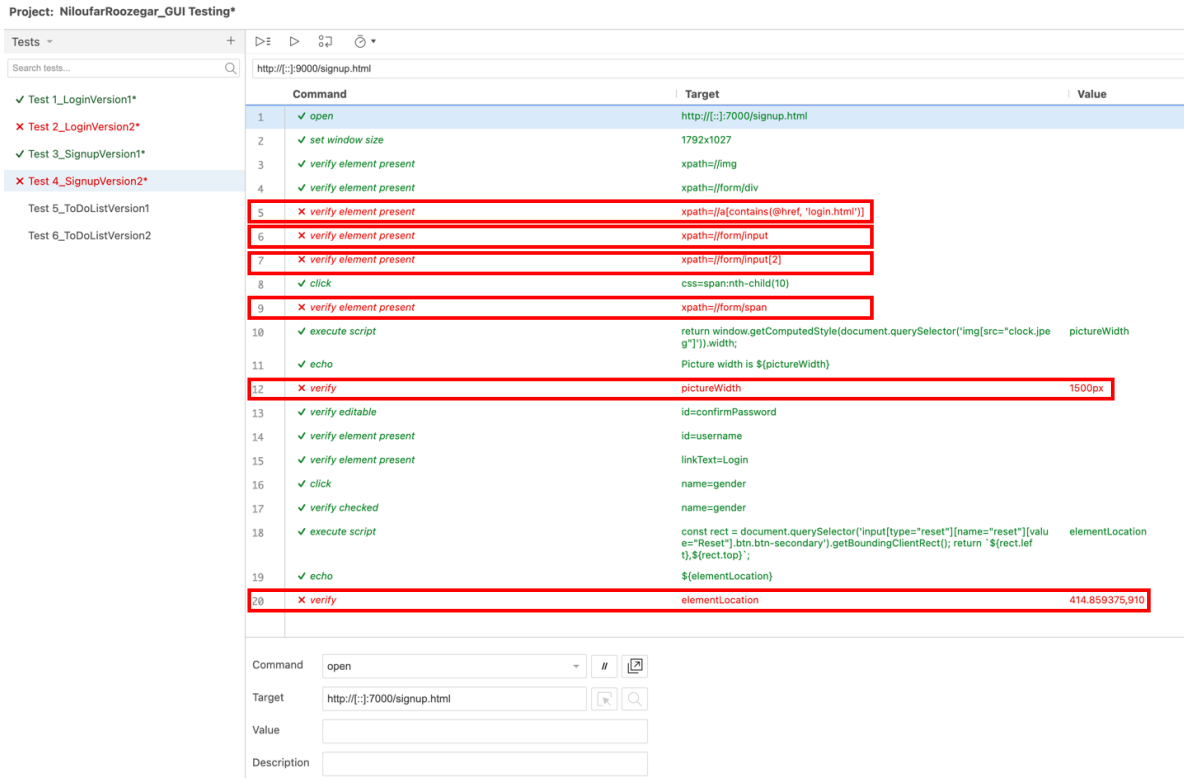
Test cases and testing results for the first and second versions of the “Signup” page are depicted in Figures 9a and 9b. Similar to the “Login” page, the first two commands were used to open the “Signup” page and set the window size. The following bullet points explain the rest of test cases created for the “Signup” page:

- Test cases number three (3) and four (4) verify the existence and position of the large picture located at the top center of the “Signup” page.
- Test case number five (5) confirms that clicking the “Login” link will redirect the user to the “Login” page, ensuring a smooth flow between the “Signup” and “Login” pages within the application.
- Test cases number six (6), seven (7), and nine (9) verify the position of the “Submit” and “Reset” buttons, as well as the position of the “Already have an account?” text.
- Test case number eight (8) check the presence of the “Already have an account?” text by clicking on the element.
- Test cases number ten (10) to twelve (12) extract the size of the picture located at the top center of the “Signup” page and verify whether the extracted image width matches the desired value.
- Test case number thirteen (13) verifies whether the password field is editable.
- Test case number fourteen (14) confirms the existence of the “Username” field, and test case number fifteen (15) verifies that the “Login” text functions like a hyperlink.
- Test cases number sixteen (16) and seventeen (17) verifies whether the radio button used for selecting the gender are checked.
- Test cases number eighteen (18) to twenty (20) extract the precise location (i.e., coordinates) of the “Reset” button on the “Signup” page and verify whether the extracted location matches the desired location.

According to Figure 9a, all the test cases passed for the initial version of the “Signup” page. However, based on Figure 9b, a total of six (6) test cases (numbers 5, 6, 7, 9, 12, and 20) failed when executed on the second version of the “Signup” page. The failed test cases are highlighted with a red box. Test case number 5 failed because clicking on the “Login” hyperlink redirected the user to the “To Do List” page instead of the intended “Login” page. In the second version of the “Signup” page, the positions of the “Submit” and “Reset” buttons, as well as the “Already have an account?” text, have been altered. Consequently, test case numbers six (6), seven (7), and nine (9), which verify the positions of these three elements, failed. Additionally, test case number twelve (12), which verifies the size of the picture located at the top center of the “Signup” page, failed due to a change in the picture's size in the second version of the application. Finally, test case number twenty (20) failed because the location of the “Reset” button was modified in the second version of the application.



(a)



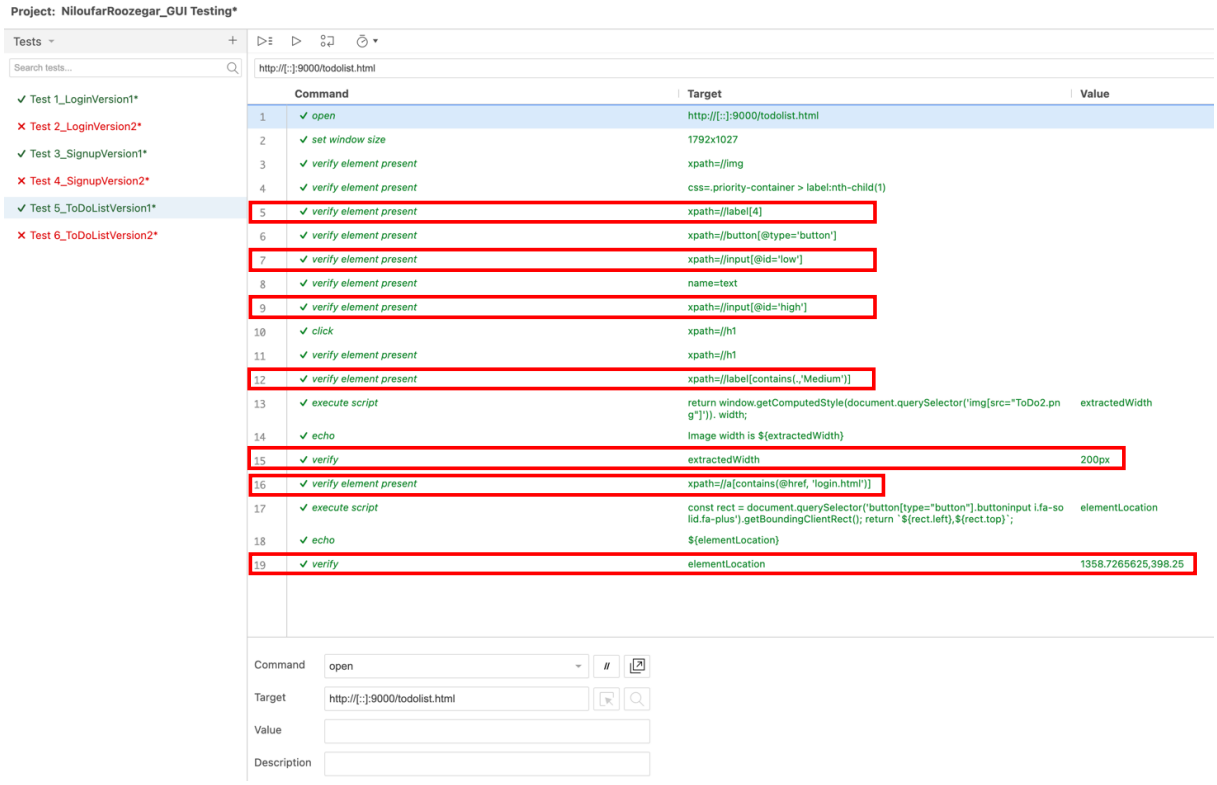
(b)

Figure 9. The test cases and results of the (a) first and (b) second version of the “Signup” page

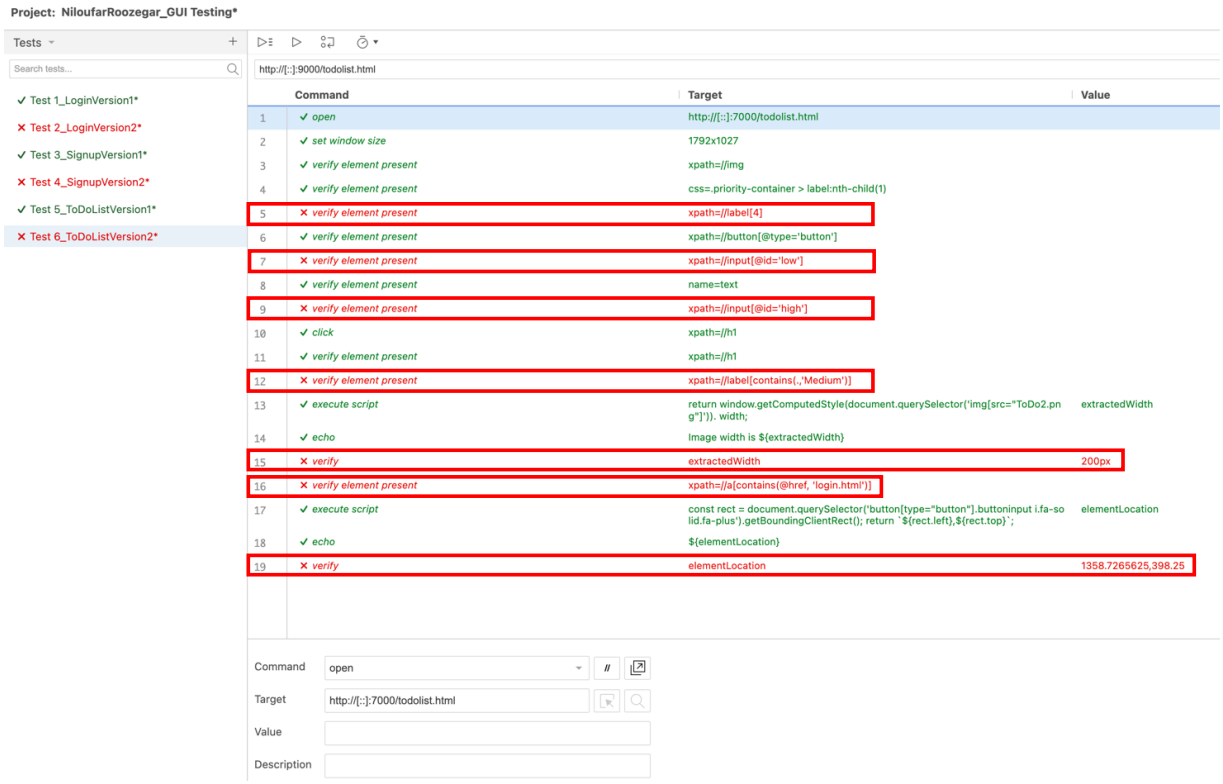
The test cases and testing results for both the first and second versions of the “To Do List” page are illustrated in Figures 10a and 10b, respectively. As with the previous pages of the application, the first two commands were employed to launch the “To Do List” page and configure the window size. The subsequent bullet points outline the remaining test cases devised for the “To Do List” page:

- Test cases number three (3) verifies the positioning of the image located at the top of the “To Do List” text.
- Test case number four (4) verifies the presence of “Priority” label on the “To Do List” page.
- Test case number five (5) verifies the position of the “High” priority radio button on the “To Do List” page.
- Test cases number seven (7) and nine (9) verify the type of element used for selecting the “Low” and “High” priority levels on the “To Do List” page.
- Test case number eight (8) checks the existence of the text field used for entering tasks.
- Test case numbers ten (10) and eleven (11) confirm the presence of “To Do List” text at the top of the “To Do List” page.
- Test case number twelve (12) verifies the presence of the “Medium” label for the priority level.
- Test cases numbered thirteen (13) to fifteen (15) involve extracting the size of the image positioned at the top center of the “To Do List” page. These test cases aim to verify whether the extracted image width aligns with the desired value.
- Test case number sixteen (16) confirms that clicking the “Sign Out” hyperlink will redirect the user to the “Login” page. This is to ensure that there exists a flow between the “To Do List” and “Login” pages within the application.
- Test cases numbered seventeen (17) to nineteen (19) involve extracting the exact location (i.e., coordinates) of the “Plus” button, which is utilized to add tasks to the “To Do List”, and verifying whether the extracted location matches the desired location.

All the test cases passed for the first version of the “To Do List” page (refer to Figure 10a). However, based on Figure 10b, a total of seven (7) test cases (numbers 5, 7, 9, 12, 15, 16, and 19) failed when executed on the second version of the “To Do List” page. The failed test cases are marked with a red box. Test case number 5 failed because the position of the High priority button changed in the second version of the application. Test cases number seven (7) and nine (9) failed because the radio buttons used for selecting the priority level in the second version of the “To Do List” page were replaced with a list of different priority levels. Similarly, test case twelve (12) failed as the label “Medium” priority does not exist in the second version of the application. Test case number fifteen (15), which verifies the size of the picture located at the top center of the “To Do List” page, failed due to a change in the picture's dimensions in the second version of the application. In the second version, clicking on the “Sign Out” button redirects the user to the “Signup” page instead of the “Login” page. This alteration led to the failure of test case number sixteen (16). Lastly, test case number nineteen (19) failed because the location of the “Plus” button was modified in the second version of the application.



(a)



(b)

Figure 10. The test cases and results for the (a) first and (b) second versions of the “To Do List” page

Assessment of the Tool

Selenium IDE is a powerful and widely used test automation tool for functional testing, which can be installed as a browser extension. This tool can generate test cases and execute the tests. It also supports multiple programming languages and provides a range of commands and assertions to validate the functionality of web elements. This part of the report presents an evaluation of Selenium IDE based on its performance in this project.

Usability

The objective of this section is to evaluate the usability of Selenium IDE in terms of Learnability, Memorability, Generating Errors, Efficiency, and Subjective Satisfaction. Installing the Selenium IDE on my Google Chrome was relatively easy and quick. The simple user interface and intuitive recording feature allowed me to create test cases without having to make extensive coding efforts. This characteristic makes the Selenium IDE an ideal choice for those who are not expert in the field. It was easy to make sense of the commands that the tool uses for generating test cases. In addition, the Selenium IDE documentations (Selenium IDE, 2018) comprehensively defines all the commands and arguments (i.e., high Learnability). Once I became familiar with the tool, it was easy to recall and reuse the acquired knowledge to generate similar test cases for other sections of the application. This is primarily due to the fact that the tool's features and functions are designed in a consistent manner (i.e., high Memorability).

Furthermore, Selenium IDE generates highly detailed error messages through its Log. Personally, I found the error messages to be very helpful as they significantly facilitated the process of identifying and fixing the errors (i.e., productive error messages). Consequently, this can enhance the efficiency of the tool. In addition, Selenium IDE effectively generates and executes various test cases by automating repetitive tasks. Test execution was fast enough that I could quickly amend the tests in cases of unsatisfactory results (i.e., high efficiency). Additionally, the user can also control the speed of test

execution. Selenium IDE offers a user friendly interface, robust functionality, error handling capabilities, a memorable workflow, and an easy-to-learn environment. Overall, I am satisfied with the tool's performance, and I would recommend it as a helpful automotive GUI testing tool.

Coverage

During the recording phase, I was able to capture all the necessary elements that required testing. Furthermore, I had the flexibility to assign specific types of testing by simply right-clicking on the desired element. In the subsequent playback phase, the recorded test steps were executed on the application under examination. This powerful GUI testing tool provided me with the ability to generate a wide range of test cases, allowing for the verification of various functionalities. In this project, I assessed the application's ability to input data into a text field and checked the functionality of radio buttons, checkboxes, and lists. The presence of various GUI elements, such as pictures, text fields, and different types of buttons, was verified through the test cases. Additionally, the tool has successfully generated automotive and semi-automotive test cases to evaluate the position, size, and precise location of the GUI elements. As can be seen in Figures 8 through 10, some of the test cases, such as those that verify the size and coordinates of the elements, were generated with the aid of JavaScript code. Furthermore, one of the main objectives of this project was to test the flow between different pages of the application. To this end, test cases were created to verify the reference links of the elements using the functionalities embedded in the "Target" section of the test cases. All in all, considering the capabilities of Selenium IDE in generating various types of test cases, it can be concluded that the tool satisfactorily met all the project objectives and performed well in terms of coverage.

Tool Features

This section of the report provides a more in-depth explanation of the tool's features. Therefore, generic information about the user interface will not be reiterated here (e.g., indicating the record and playback buttons). As shown in Figure 11, upon opening the application, I was able to right-click on any

element to assign the desired type of testing. Once the recording was completed, the tool automatically generated all the test cases assigned to various elements. Each test case consists of three sections: Command, Target, and Value. The Command can be selected during the recording phase, while the objective of the test can be modified by clicking on the Target box. As illustrated in Figures 12a, the position of the “Reset” button can be tested by selecting the "xpath:position" option. Additionally, Figure 12b depicts how the tool can verify whether the hyperlink “Login” redirects the user to the “Login” page.

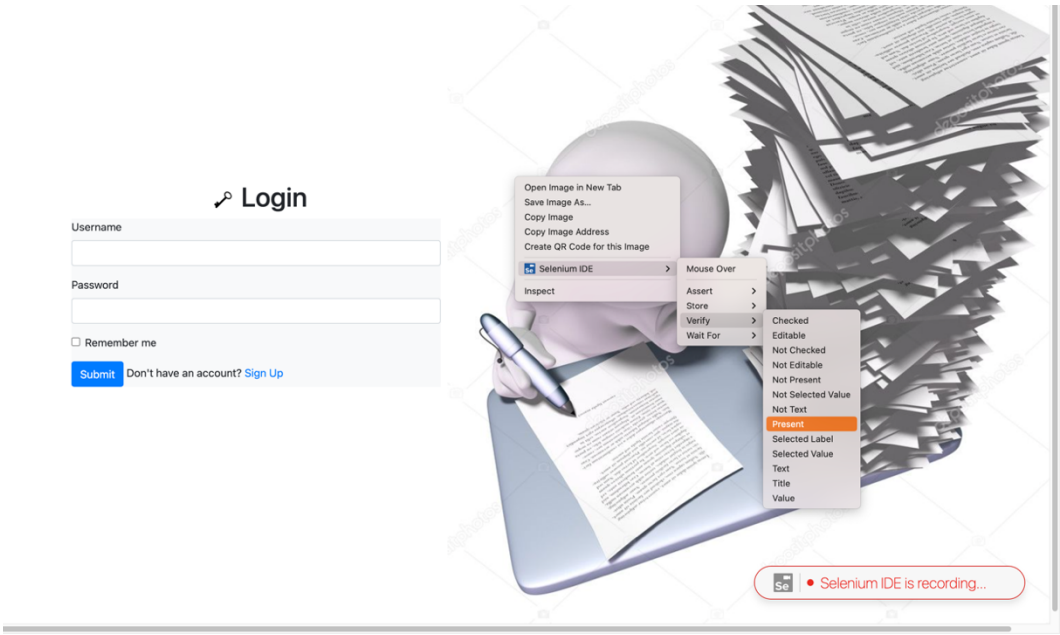
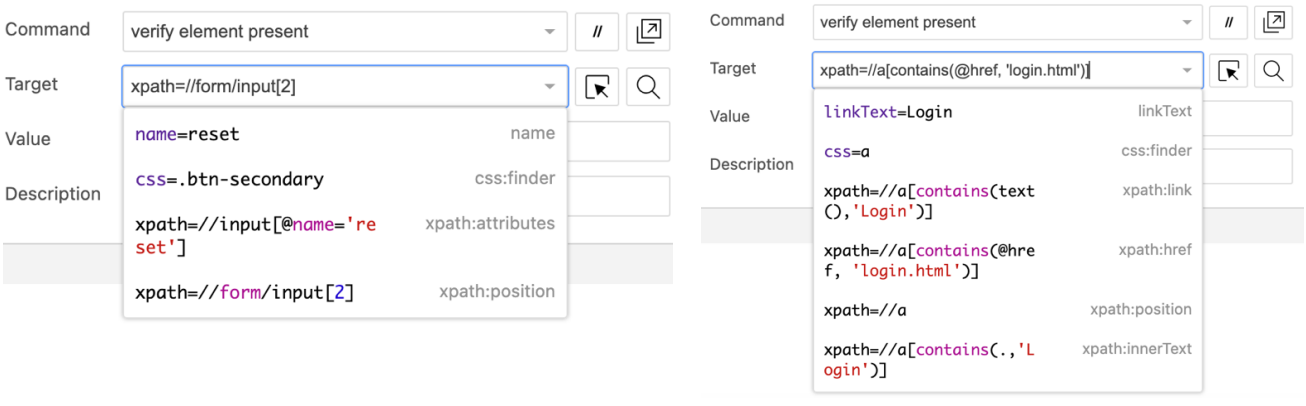


Figure 11. Assigning Various Types of Tests



(a) (b)

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

To satisfy the objectives of this project, the size and the location of different elements need to be tested. A semi-automated method was employed, utilizing codes written in “Execute Script”, to extract the size and coordinates of the elements, and subsequently comparing the extracted values with the desired values through the "Verify" command. The codes used to test the size and location of the elements are depicted in Figures 13a and 13b.

✓ *execute script*	return window.getComputedStyle(document.querySelector('img[src="loginkey.jpeg"]')). width;	extractedWidth
✓ *echo*	Image width is ${extractedWidth}	
✓ *verify*	extractedWidth	40px

(a)

✓ *execute script*	const rect = document.querySelector('input[type="submit"][name="submit"][value="Submit"].btn.btn-primary').getBoundingClientRect(); return `${rect.left},${rect.top}`	elementLocation
✓ *echo*	${elementLocation}	
✓ *verify*	elementLocation	333.5,589.359375

(b)

Figure 13. The methods of testing the (a) size and (b) location of the elements

Evaluation of the Reusability of Test Cases and Final Results

One of the main goals of this project was to generate sets of test cases for the initial version of the application and subsequently reuse similar test cases to evaluate the second version of the application. This was easily accomplished using Selenium IDE. Initially, the set of test cases developed for the first version of the application was duplicated. Subsequently, the “Open” command was modified to open the second version of the application. Through this approach, the same test cases that were generated for the first version of the application were successfully reused to test the second version as well.

According to the project specifications, a certain number of changes had to be applied to the second version of the application. As the same sets of test cases were utilized for the second version of the application, it was expected that some of the test cases involving these changes would fail. The test

results of the second version indicated that all the test cases targeting the elements subjected to changes did indeed fail, while the remaining test cases examining unchanged elements passed. These results demonstrate that Selenium IDE successfully detected all the major changes in size, position, and location of the elements, as well as the flow between different pages of the application. Additionally, the outcomes imply that Selenium IDE is a reliable GUI testing tool, as all the test cases targeting the unchanged elements were passed.

References

Qi, S. (2020). *Evaluation of the maintenance required by web application test suites* (Doctoral dissertation, Politecnico di Torino).

Selenium IDE. (2018). *Documentation*. https://www.selenium.dev/selenium-ide/docs/en/introduction/getting-started

Vaswani, G. A. (2022). *A blockchain-based approach for securing Electronic Hospital Records* (Doctoral dissertation, Dublin, National College of Ireland).