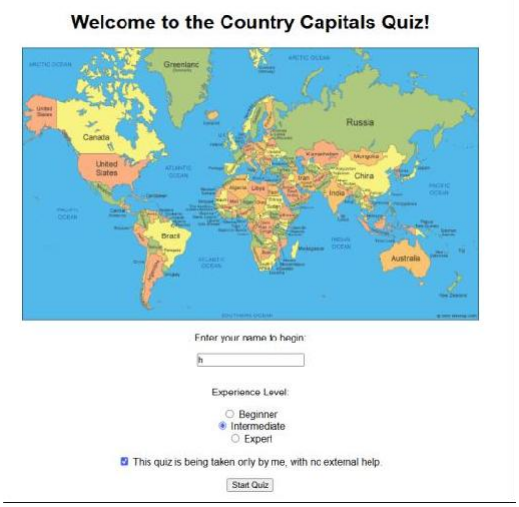
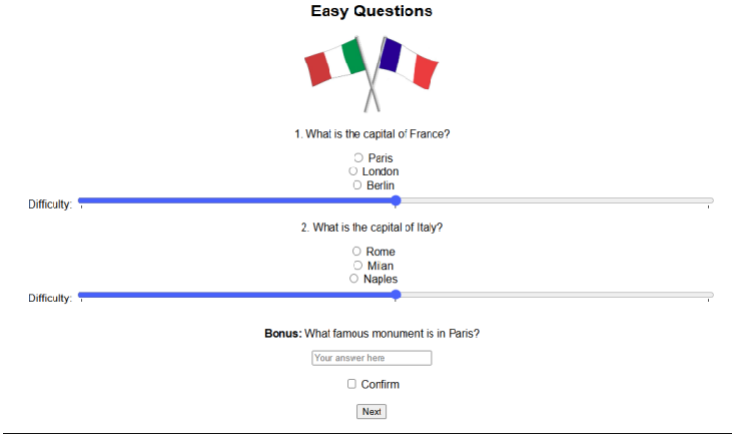
# Graphical User Interface Testing Assignment

## Version 1



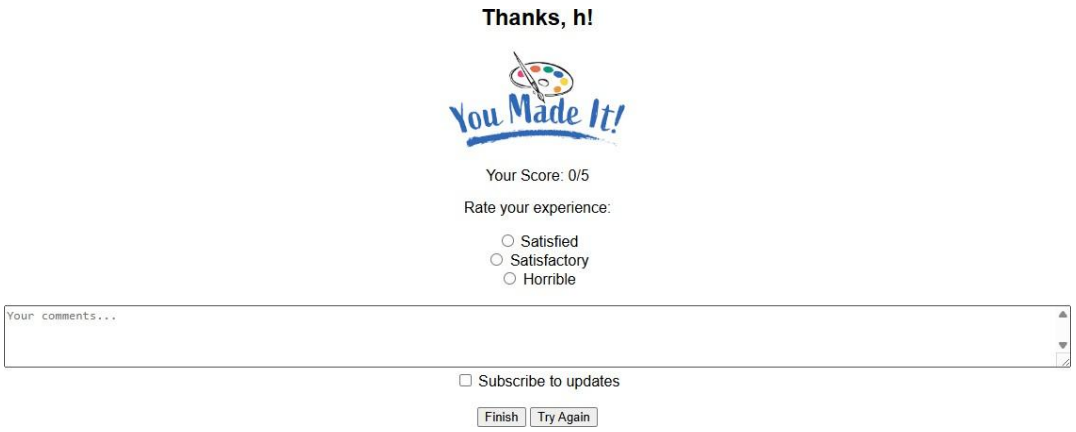

The first screenshot is of the first page of the quiz application where there is an image of the world, the user must enter their name into the textbox, select one of the radio buttons to specify their experience, select the checkbox to honor the quiz and then click the button to start the quiz.




The second, third and fourth screenshot are similar to each other they are of easy, medium and hard questions with an image of the respective countries of which the questions are, the user must select a radio button to answer the question. Then they can fill in the textbox for the bonus question and select the checkbox to confirm their answers and move on to the next set of questions.



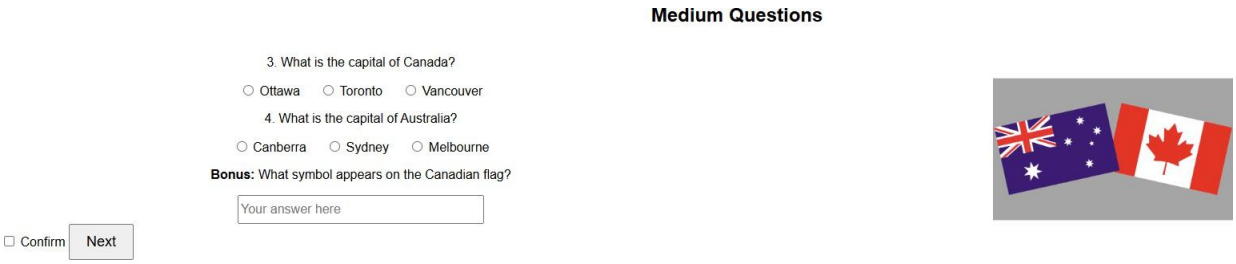
As stated above the third and fourth screenshots are similar to the second screenshot but in the fourth screenshot after selecting the checkbox to confirm the answers the user can select the submit button to end the quiz and move on to the results page.



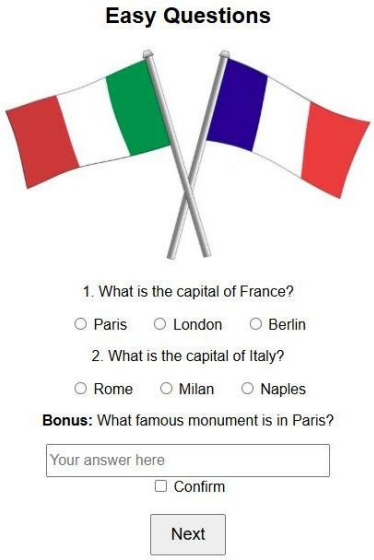

The fifth screenshot is where the user can select the radio button to rate their experience, a text area to provide comments, they can select the checkbox to subscribe to updates and click finish button to end the quiz or try again button which will take them to page 1. The image as a good gesture.

## Version 2


In the first screenshot the **image size has been reduced**, The **text box** to enter the users name **has been made bigger**, the **orientation of radio buttons** has been switched from vertical **to horizontal** and the start quiz **button size** has been **increased** as well.




In the second screenshot we can see that the **flow of pages has been changed**, medium questions page has come before easy questions as opposed to how it was in version 1. The **image is on the right side** of the page instead of center, the **size of text box** to enter bonus answer and **next button** have been **increased**. The **checkbox and button** have been **placed on the left**. The **orientation of radio buttons** has been **changed to horizontal** as well.




In the third screenshot we see that the **image size** has been **increased**, the **radio buttons orientation** has been **changed**, the **size of text box** to enter bonus answer and the **next button** has also been **increased.**


In the fourth screenshot the **image is placed on the left** and everything else on the right as opposed to center in Version 1. The **size of textbox** and **submit button** have been **increased** and the **radio buttons orientation** has been **changed to horizontal** as well.


In the final screenshot the **image size has been increased**, the **text area size** has been **reduced**, and the **radio buttons orientation** has been **changed.**

The tool used for testing is Selenium. It is one the most powerful and flexible options for testing GUI components. It is an open-source automation tool that allows users to simulate interactions like clicking buttons, navigating through webpages and entering text making it very useful and effective for browser-based GUI tests. It's not a visual testing tool but is excellent for functional GUI testing using its wide variety of features. It enables users to create regression tests, functional tests and other types of GUI tests.

**Components of Selenium**

Selenium is a suite of tools that comprise Selenium IDE, Selenium WebDriver, Selenium RC and Selenium Grid.

Selenium IDE is an open-source integrated development environment used for conducting web testing and automation.

Selenium RC came before the WebDriver and facilitated interaction with various browsers using a server that was like a middleman between the code and browser.

Selenium WebDriver unlike RC directly communicates with the browsers native support for automation making it more reliable and stable for testing

Selenium Grid is a server that allows tests to be run in parallel and helps in managing different versions of the browser

**What is possible with selenium**

Functional Testing – It validates that the application is behaving as it should.

Regression Testing – It sees to it that new code changes haven't broken the existing features. Selenium scripts can be reused to test the same flows.

Data-Driven Testing – It has the capability to integrate with Excel, CSV and databases to run the same test with multiple inputs. This deems useful when testing is to be done on forms and search features.

CI/CD Pipelines – It can work with tools like Jenkins, GitHub and Azure DevOps for automated test execution ensuring that testing happens automatically when new code has been deployed.

**Test Cases to verify the existence of GUI elements**

Verifying if GUI elements are present on the webpage is very crucial as these elements help the user have a seamless experience and in turn make the webpage useful. If these elements are not rendered then the purpose of the web application becomes null and void. The test cases written to verify the existence are:

- testImageExists1()
- testImageExists2()
- testImageExists3()
- testImageExists4()
- testImageExists6()
- testTextboxExists1()
- testTextboxExists2()
- testTextboxExists3()
- testTextboxExists4()
- testRadioButtonsExists1()
- testRadioButtonsExists2q1()
- testRadioButtonsExists2q2()
- testRadioButtonsExists3q3()
- testRadioButtonsExists3q4()
- testRadioButtonsExists4()
- testRadioButtonsExists6()
- testCheckboxExists1()
- testCheckboxExists2()
- testCheckboxExists3()
- testCheckboxExists4()
- testCheckboxExists6()
- testButtonExists1()
- testButtonExists2()
- testButtonExists3()
- testButtonExists4()
- testButtonExists6b1()
- testButtonExists6b2()

## Test Cases to verify the Location and Size of GUI elements

Verifying the location and size of GUI elements is very crucial, GUI elements are not subject to change in a webpage unless manually done. So, whenever these tests are run it should verify that the location of the GUI element has not changed. The size helps in verifying that it is of non-zero size ensuring visibility. The test cases written to verify the location and size are:

- testImageLoc1()
- testImageLoc2()
- testImageLoc3()
- testImageLoc4()
- testImageLoc6()
- testImageSize1()
- testImageSize2()
- testImageSize3()
- testImageSize4()
- testImageSize6()
- testTextboxLoc1()
- testTextboxLoc2()
- testTextboxLoc3()
- testTextboxLoc4()
- testTextboxSize1()
- testTextboxSize2()
- testTextboxSize3()
- testTextboxSize4()
- testRadioButtonsLocAndSize1()
- testRadioButtonsLocAndSize2q1()
- testRadioButtonsLocAndSize2q2()
- testRadioButtonsLocAndSize3q3()
- testRadioButtonsLocAndSize3q4()
- testRadioButtonsLocAndSize4()
- testRadioButtonsLocAndSize6()
- testCheckboxLocAndSize1()
- testCheckboxLocAndSize2()
- testCheckboxLocAndSize3()
- testCheckboxLocAndSize4()

- testCheckboxLocAndSize6()
- testButtonLocAndSize1()
- testButtonLocAndSize2()
- testButtonLocAndSize3()
- testButtonLocAndSize4()
- testButtonLocAndSize6b1()
- testButtonLocAndSize6b2()

**Test Cases to verify the content of the GUI elements**

In this we verify if the element has the correct information such as value, type or text. This helps in maintaining a web application to the standard it needs to provide and serves its purpose. This is very critical for form elements, buttons and controls. The test cases written to verify the content are:

- testImageContent1()
- testImageContent2()
- testImageContent3()
- testImageContent4()
- testImageContent6()
- testTextboxContent1()
- testTextboxContent2()
- testTextboxContent3()
- testTextboxContent4()
- testRadioButtonsContent1()
- testRadioButtonsContent2q1()
- testRadioButtonsContent2q2()
- testRadioButtonsContent3q3()
- testRadioButtonsContent3q4()
- testRadioButtonsContent4()
- testRadioButtonsContent6()
- testCheckboxContent1()
- testCheckboxContent2()
- testCheckboxContent3()
- testCheckboxContent4()
- testCheckboxContent6()
- testButtonContent1()

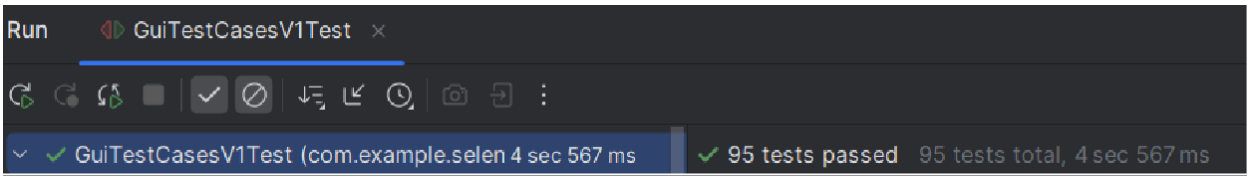
- testButtonContent2()
- testButtonContent3()
- testButtonContent4()
- testButtonContent6b1()
- testButtonContent6b2()

**Test Cases to verify flow from one page to another**

Navigation flow from one page to another is very important in a web application. It concerns user experience and engagement. These tests ensure that the next webpage to load is of the correct URL and there is no disruption in the how the web application works. The test cases written to verify the flow are:

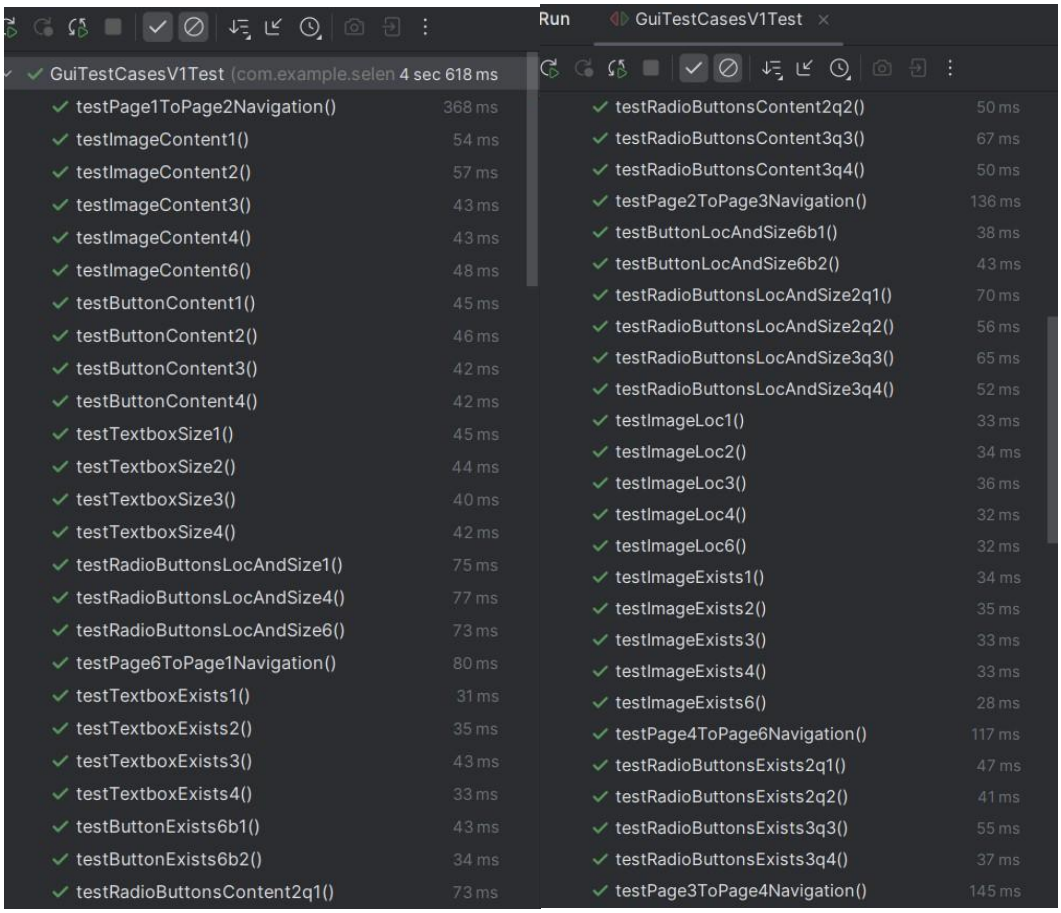
- testPage1ToPage2Navigation()
- testPage2ToPage3Navigation()
- testPage3ToPage4Navigation()
- testPage4ToPage6Navigation()
- testPage6ToPage1Navigation()

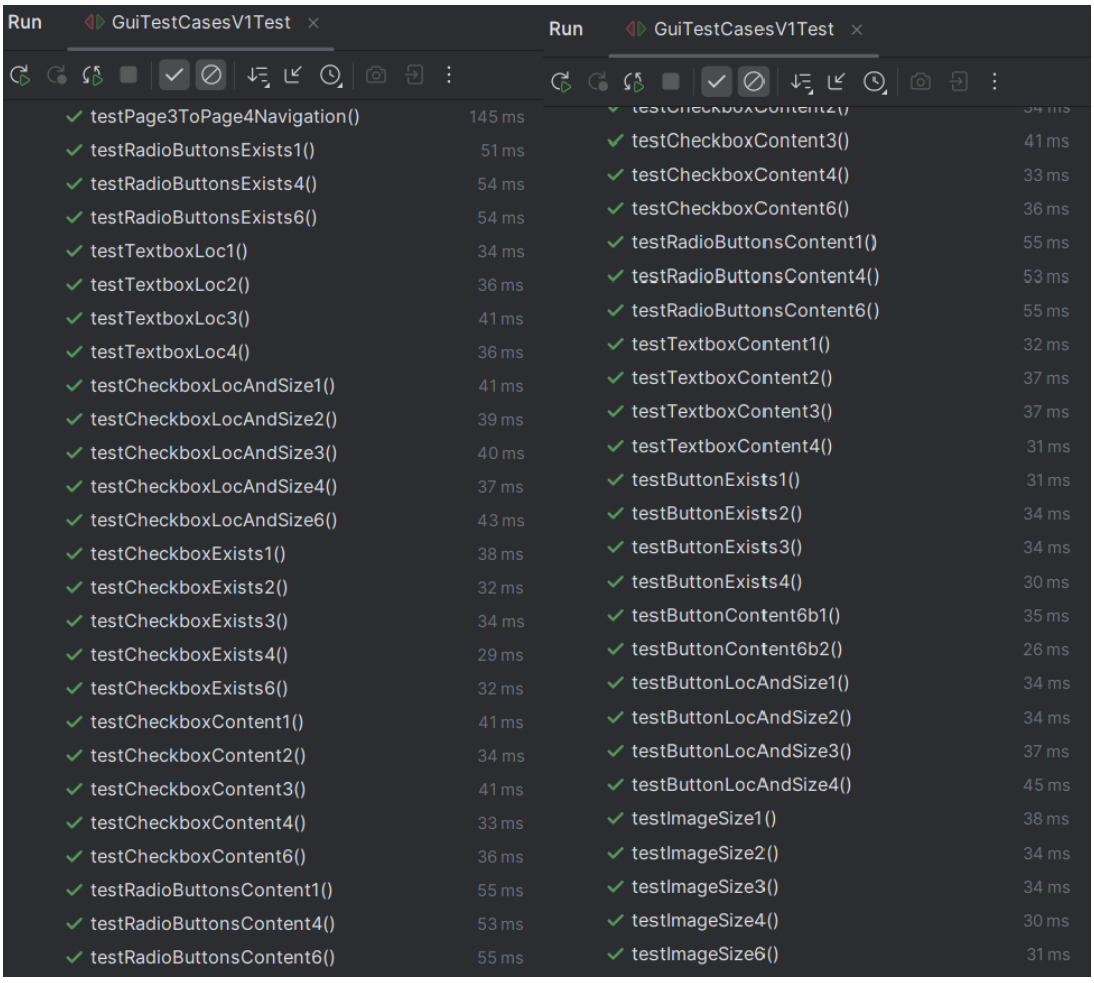
## Screenshots of Test Cases for Version 1



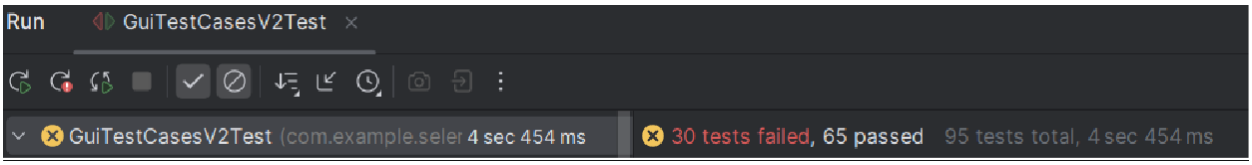

As we can see all the test cases written to check for existence, location, size, content and flow of the pages have passed.

Below are the screenshots of the 95 test cases that verify the above.



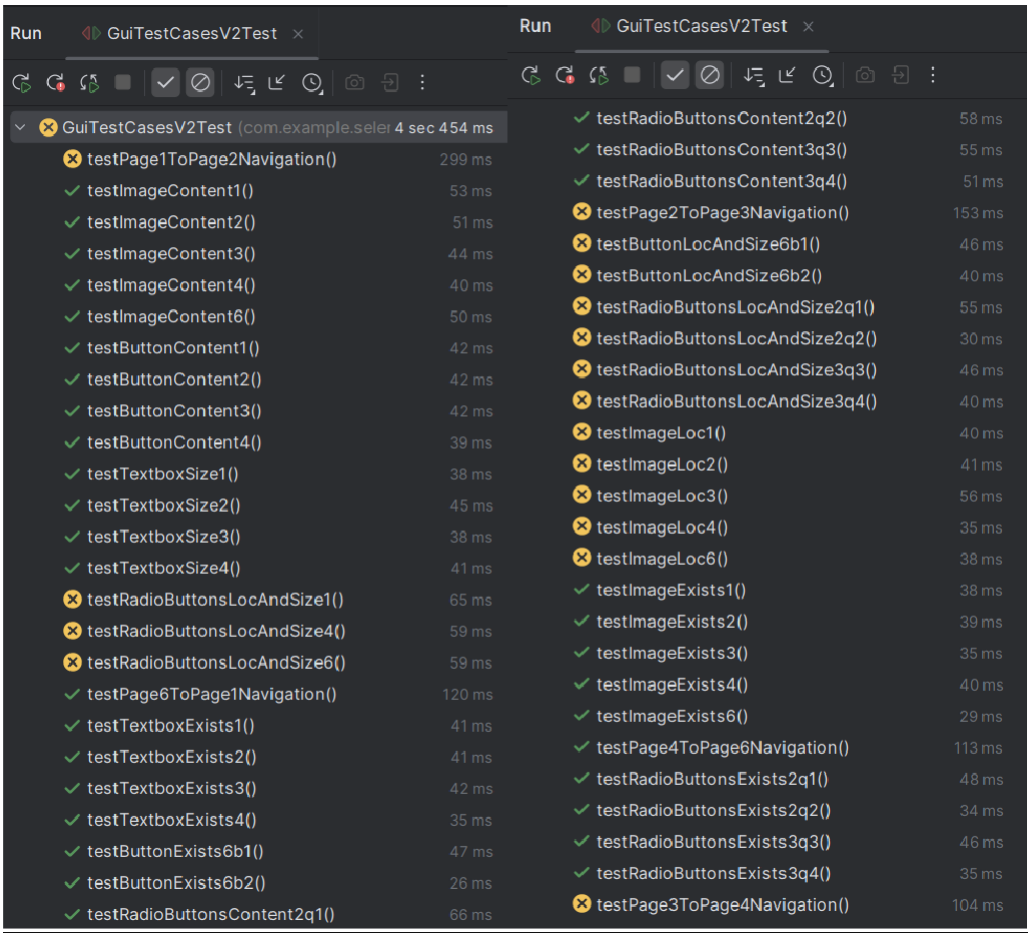


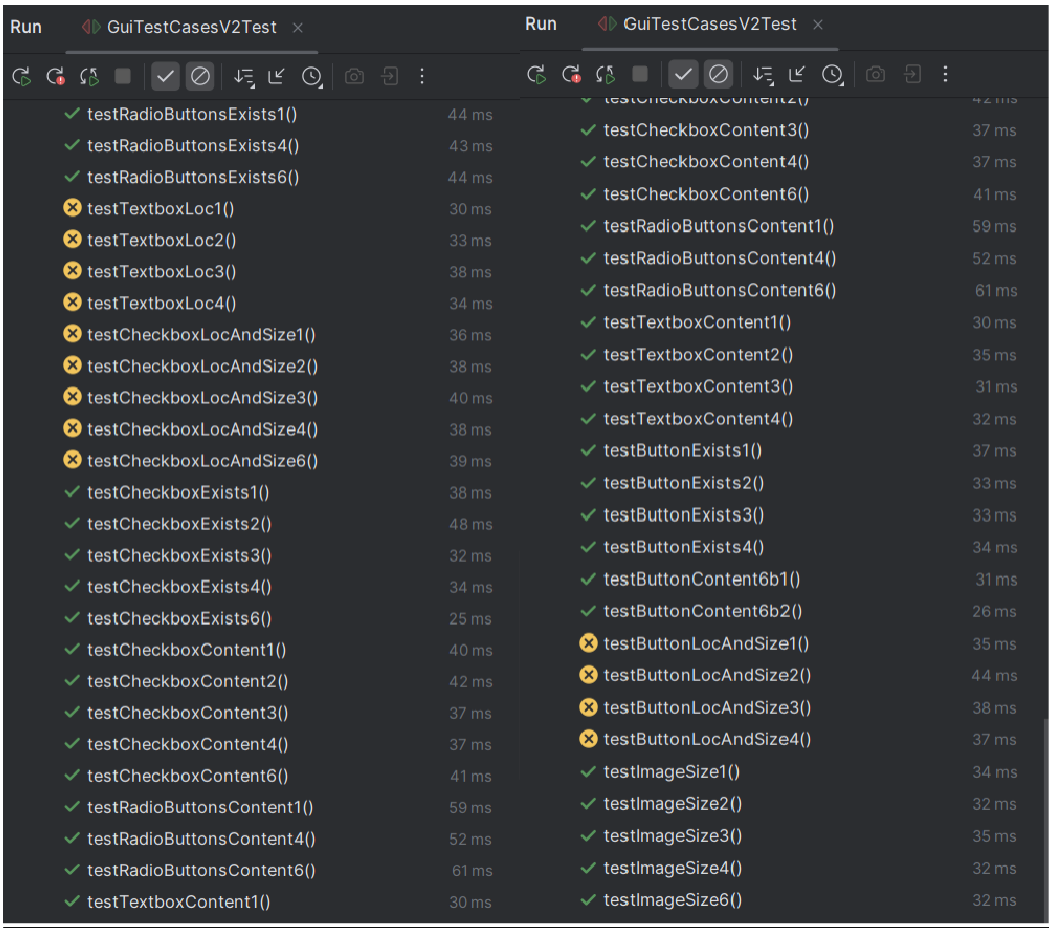

Now we move onto executing the same testcases on version 2 of the web application




As the test cases were designed specifically for version 1 of the GUI, it is bound to fail for version 2 in some cases. Hence 30/95 test cases have failed and 65/95 have passed.

The below screenshot give us an idea of which test cases passed and failed.






We can see 3 types of test cases that have failed. Firstly, the **navigation flow test case**, flow of pages was to be altered and hence there is bound to be a mismatch in flow between 3 pages for one change hence the failure in navigation test cases.

Secondly, the **Location of the GUI elements**, these were altered as well in version 2 of the GUI, so it only makes sense that the X and Y coordinates of these GUI components do not match the given value in the test cases.

Thirdly, the **Size of the GUI elements**, various elements like images, textboxes and buttons were modified with respect to its size, some enlarged and some diminished.

## Features

- **Support for Multiple Programming Languages:** Tests can be written in multiple languages not limited to Java and Python, but JavaScript, Ruby, C# and others as well.
- **Integration:** Selenium integrates well with CI/CD pipelines and other testing tools and it supports automated regression and functional testing as well. It can be integrated with frameworks like Junit and TestNG as well for scalable and automated testing
- **Cross-Browser Compatibility:** Selenium works with most of the major web browsers like Google Chrome, Mozilla Firefox, Microsoft Edge and Safari as well.
- **Cross-Platform Compatibility:** It supports multiple operating systems such as Windows, macOS and Linux.
- **Parallel Execution:** Selenium Grid allows tests to run in parallel across multiple browsers and environments making it fast and efficient.
- **Functional and Regression Testing:** Selenium can be used for both functional and regression testing.
- **Simulates Interactions:** It can replicate user behaviors, such as mouse movements, keyboard inputs and drag-and-drop actions, which helps us test the GUI thoroughly.
- **WebDriver:** This helps in providing APIs to control the specific browser and the whole interaction process with web elements.

## Type of Coverage

- Functional Coverage - Functional Testing where in button clicks, form submissions, dropdowns, and navigation flows are tested.
- Multiplatform coverage - Verifies UI rendering and behavior across multiple bowsers and operating system.
- Responsive Design Coverage – Tests different layouts on different screen sizes via cloud platforms. It provides adaptive layout testing.

## Reuse of Test Cases

Selenium facilitates extensive test case reuse through several approaches. Test scripts can be written in a very modular manner where they can be reused across various test cases. It works with Junit and PyTest as well helps in managing suites, fixtures and reusability. It offers parameterized test cases as well. It helps in component -level reuse across multiple pages as few tests are the same no matter which webpage it is (Same reason V2 passed 65 test cases from V1). The high, test reusability that selenium offers helps in better maintenance of code and saves time as well.

## Test Results Produced

Selenium tests can produce various results be it, pass or fail status of test cases, execution time metrics (step by step and as whole), exception details and error messages. But it doesn't provide reports by itself, this is possible only through integration with frameworks like TestNG and Junit or any other custom reporting framework.

## Ease of Usage

- It is a very beginner friendly tool, it has a strong IDE support and good debugging capabilities.
- The WebDriver does require previous knowledge of a programming language, but it can be any of the popular languages like Java, Python, JavaScript, etc.
- Selenium has a big ecosystem and forum, easy to find solutions and information on Stack Overflow and GitHub.
- It easily integrates with Maven or Gradle making the process of automation very seamless.
- It works well with any browser be it Chrome, Edge or Firefox.

## Types of GUI elements that can be tested

It facilitates testing of different types of GUI elements like

- Buttons
- Input
- Text Areas and Textboxes
- Checkboxes and Radio Buttons
- Dropdowns, Sliders
- Links
- Forms
- Tables
- Drag and Drop elements
- Images
- Alerts and Dialog boxes

To sum it up Selenium is a robust and powerful tool to test GUI components in a web application. Its wide variety of features ranging from integration to portability are a great reason to choose this framework to test web applications.