CSE565: SVVT- Assignment 5

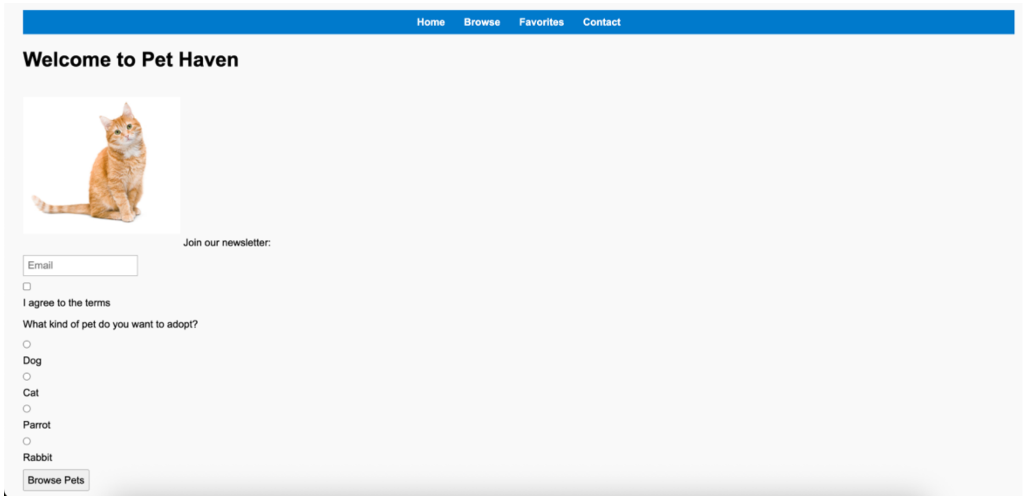
Asish Addanki
ASU ID:1230916442

1. Description of the application:

The application developed is a simple, friendly web application designed to help users explore, select, and express interest in adopting pets. The application offers a clean and intuitive graphical user interface (GUI) with multiple interactive features such as image galleries, form inputs, favorite lists, and navigation flows.

The application is the version 1, and consists of four main pages:

- Home Page: A welcoming landing page where users can view a promotional image, subscribe to a newsletter, and select the type of pet they are interested in.



Home Browse Favorites Contact

Welcome to Pet Haven

Join our newsletter:

Email

I agree to the terms

What kind of pet do you want to adopt?

Dog

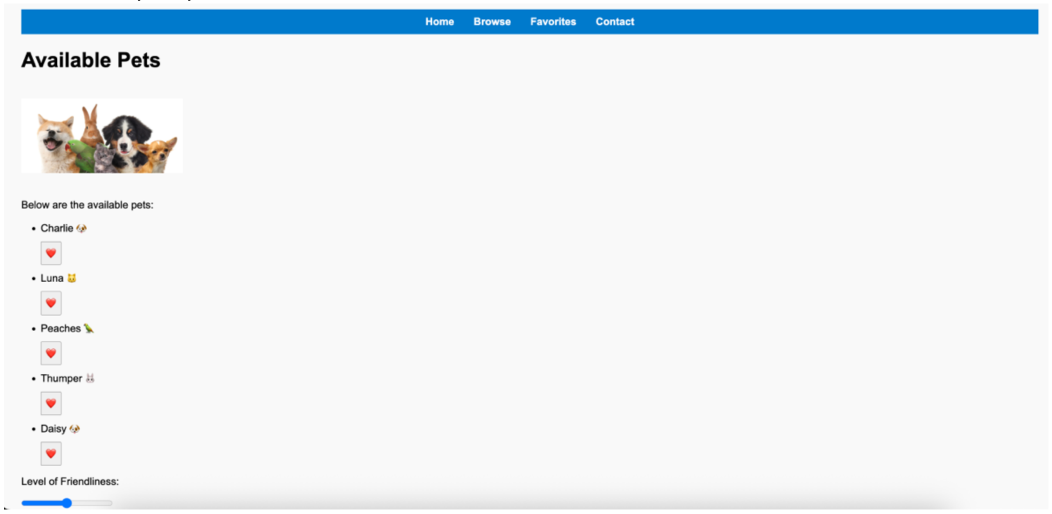
Cat

Parrot

Rabbit

Browse Pets

- Browse Page: A gallery of available pets, allowing users to mark their favorites and describe why they wish to adopt a pet.



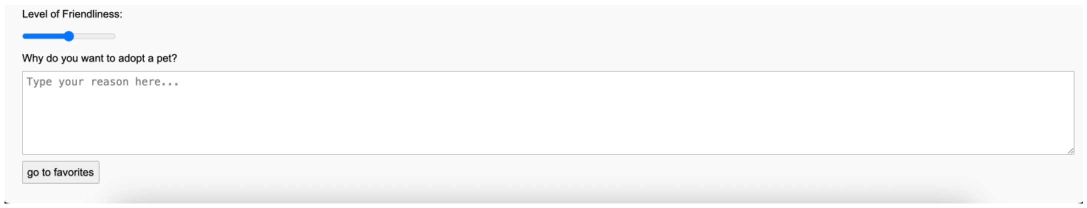
Home Browse Favorites Contact

Available Pets

Below are the available pets:

- Charlie 🐶
- Luna 🐱
- Peaches 🦜
- Thumper 🐰
- Daisy 🐶

Level of Friendliness:

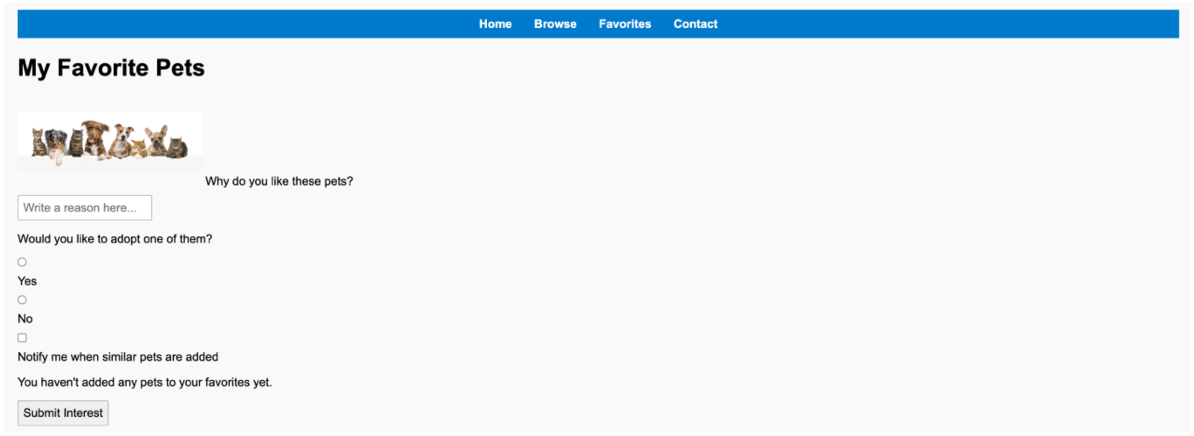

Level of Friendliness:

Why do you want to adopt a pet?

Type your reason here...

go to favorites

- Favorites Page: A personal collection of pets the user has favorited, with the ability to clear favorites and provide a rating.



My Favorite Pets

Why do you like these pets?

Write a reason here...

Would you like to adopt one of them?

Yes

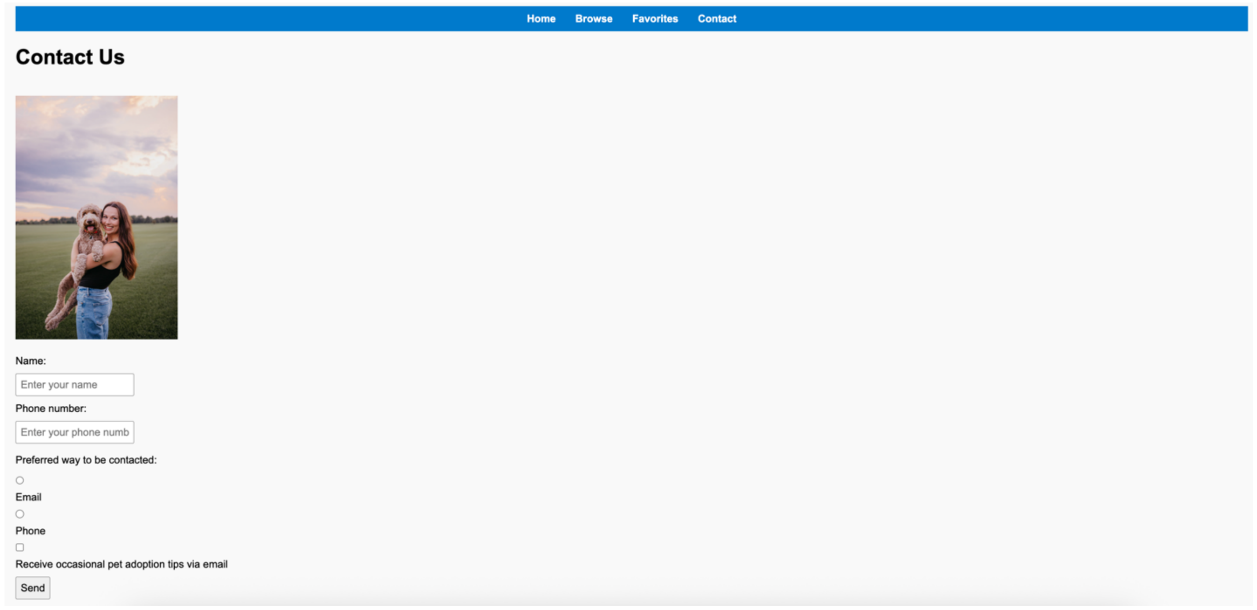
No

Notify me when similar pets are added

You haven't added any pets to your favorites yet.

Submit Interest

- Contact Page: A form to collect user information and preferred contact methods, supporting direct engagement for pet adoption inquiries.



Contact Us

Name:

Enter your name

Phone number:

Enter your phone numb

Preferred way to be contacted:

Email

Phone

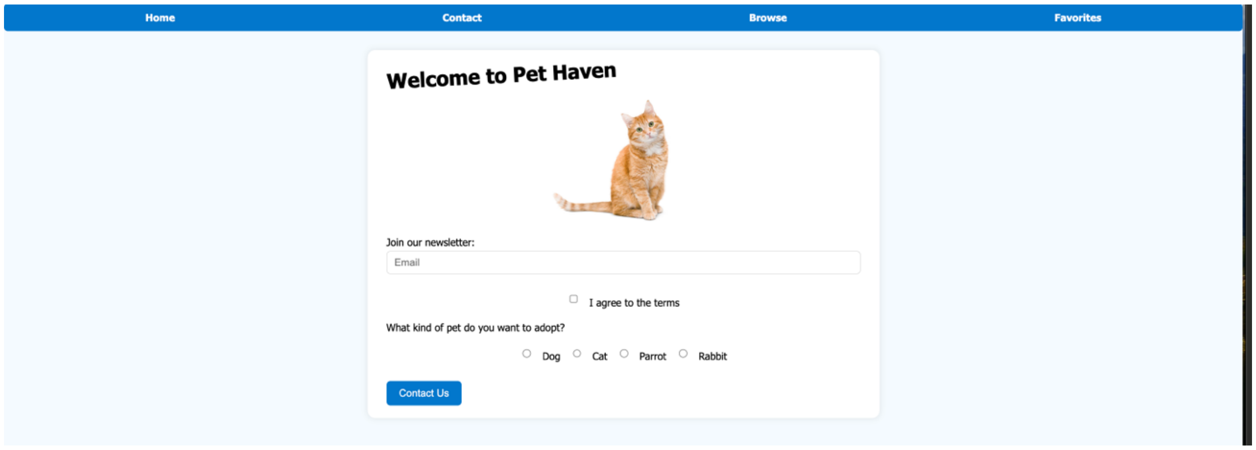
Receive occasional pet adoption tips via email

Send

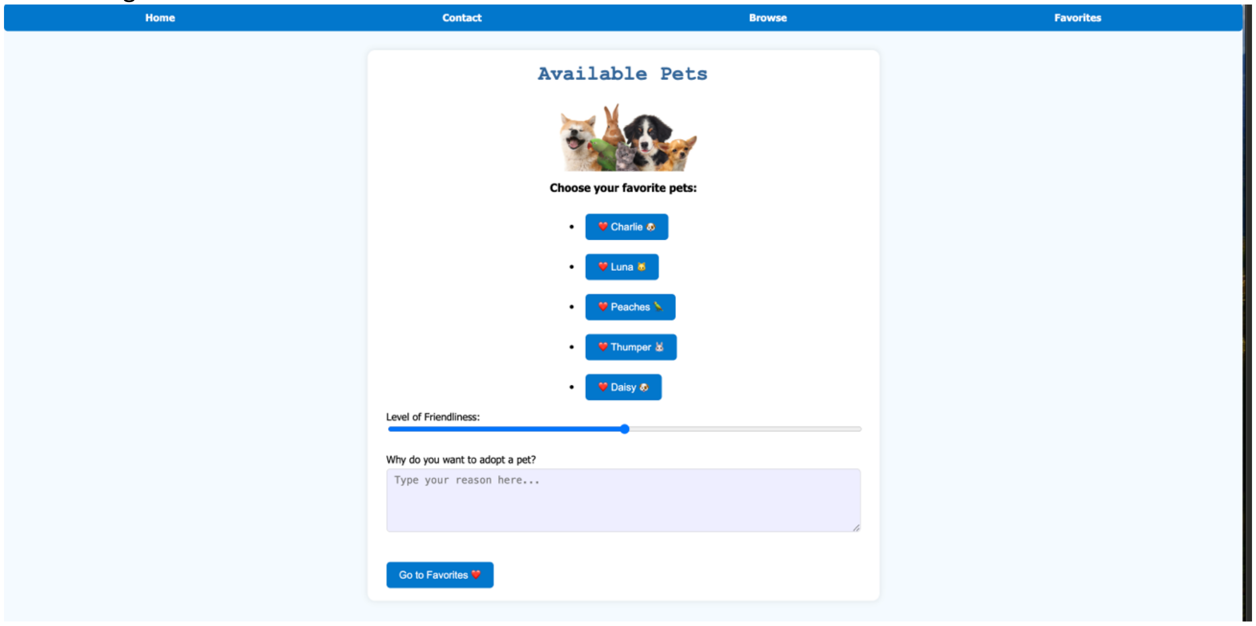
The design focuses on simplicity and user experience, with colorful images, interactive buttons, radio buttons, checkboxes, and sliders. All pages maintain a consistent navigation bar allowing seamless movement between sections.

Below are the screentshots of version 2 with the given modifications:

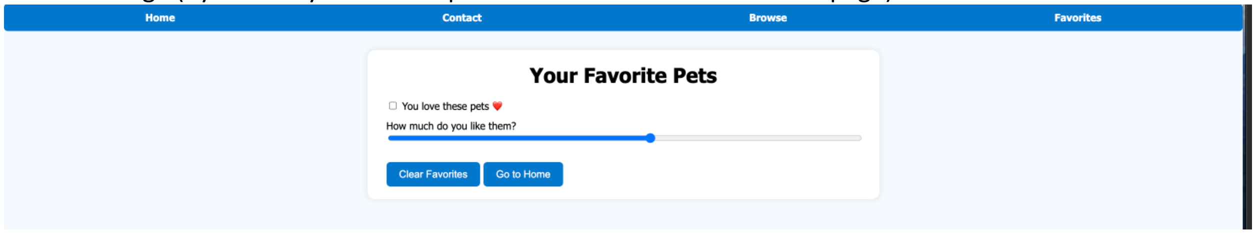
- Home Page



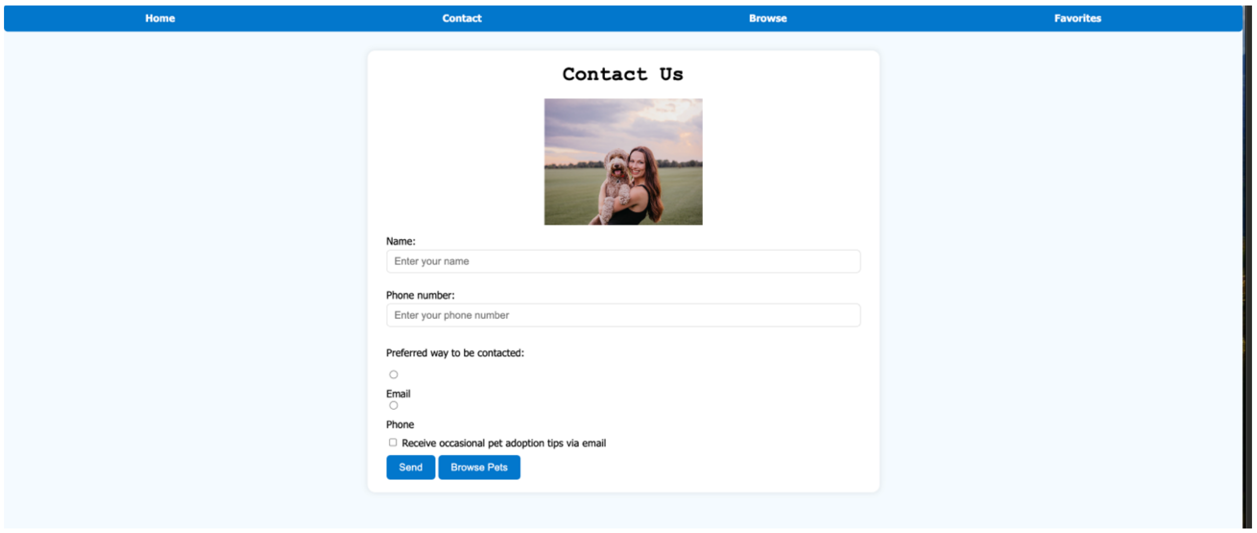
- Browse Page



- Favorites Page (dynamically shows the pets that are added from browse page)



- Contact Page



Contact Us

Name:

Enter your name

Phone number:

Enter your phone number

Preferred way to be contacted:

Email

Phone

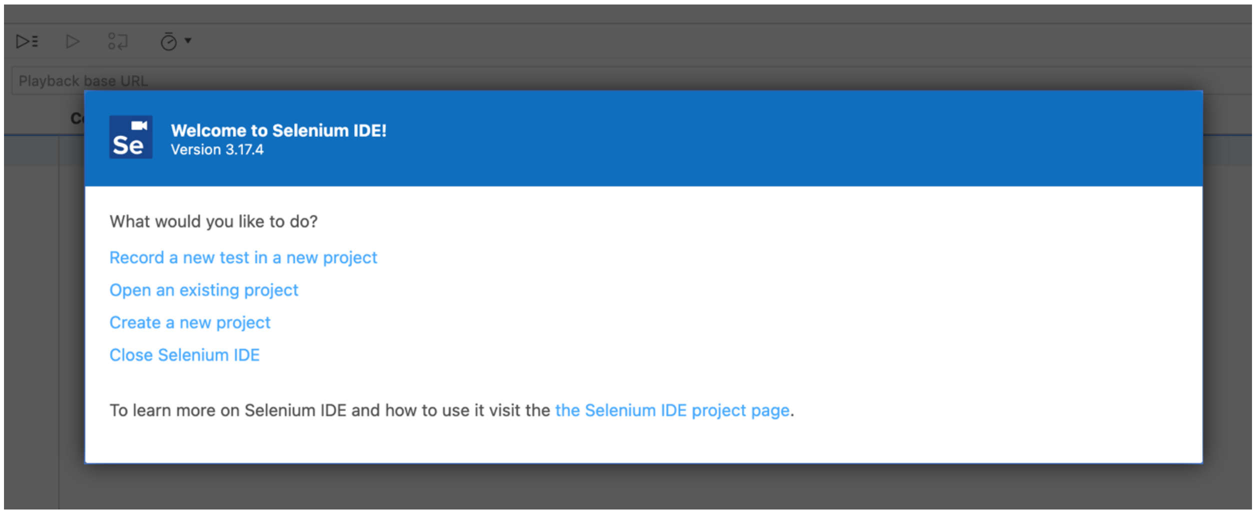
Receive occasional pet adoption tips via email

In addition to Version 1, a Version 2 of the website was developed, introducing changes in element orientation, sizing, positioning, and navigation flow to simulate a real-world GUI evolution scenario. Specific changes included repositioning buttons, adjusting the sizes and locations of input fields, modifying the order of navigation links, and slightly altering the interaction flow between pages. These intentional adjustments reflect how even small design updates can significantly impact user experience and the stability of automation scripts.

2. Description of the tool selected for GUI:

For this project, I selected Selenium IDE as the tool for GUI testing. Selenium IDE is one of the most popular open-source frameworks for automating web applications and was a natural choice because of its flexibility and strong community support. It allows testers to interact with web pages almost exactly like a real user would such as clicking buttons, entering text, selecting options, and navigating across different pages. Selenium supports Firefox , and it integrates well with Java, which made it a perfect fit for the way this project was set up. In this case, I used Selenium to automate the testing of both Version 1 and Version 2 of the Pet Haven website. The automated tests included actions like filling forms, adding favorite pets, checking for confirmation alerts, and validating the page flows. Using Selenium not only saved time compared to manual testing, but it also clearly highlighted how small changes in a GUI like moving a button or changing a page layout can break existing test scripts which was observed later in the version 2 of the test results. Overall, Selenium proved to be a very effective tool for demonstrating the impact of GUI changes on automated testing.

Once the extension is set up in the firefox, with just a click on the extension, we are prompted to open or create a new project and once the suitable option is selected, we will create the project name and make a new test case and name it and then we can hit the REC button.

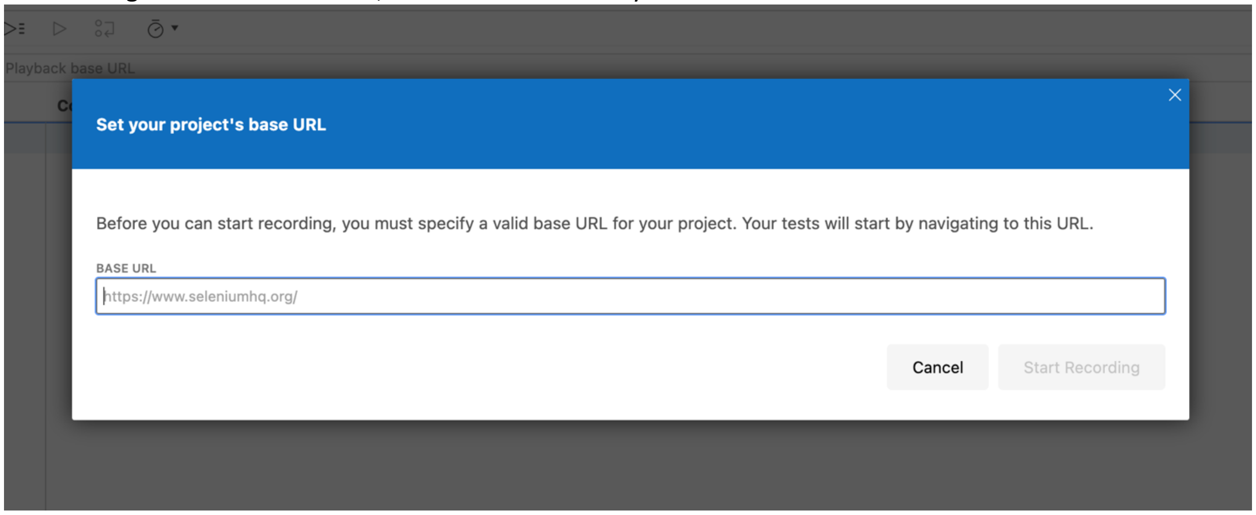

Welcome to Selenium IDE!
Version 3.17.4

What would you like to do?

Record a new test in a new project
Open an existing project
Create a new project
Close Selenium IDE

To learn more on Selenium IDE and how to use it visit the the Selenium IDE project page.

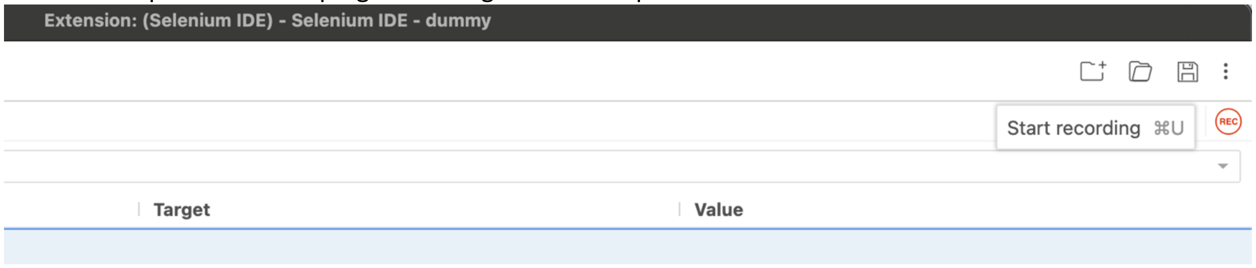
An URL must be specified before creating a new test case so that all the test cases that are in the project follow the given URL. In this case, I have hosted it locally.


Set your project's base URL

Before you can start recording, you must specify a valid base URL for your project. Your tests will start by navigating to this URL.

BASE URL
https://www.seleniumhq.org/

The record option on the top right corner gives us the option to record the test cases.



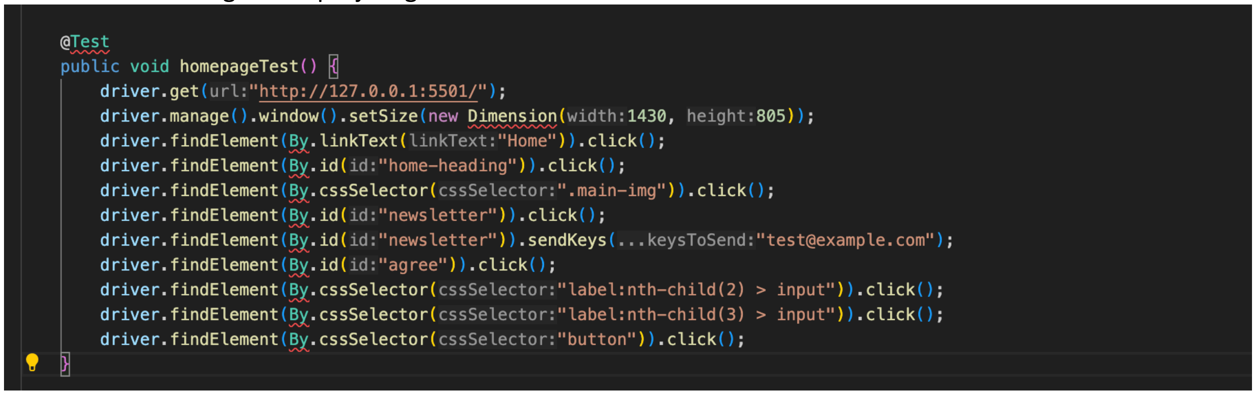
Once the "Record" button is clicked, a new tab opens in Firefox with the specified URL. From there, we interact with the page such as clicking buttons, filling in forms, selecting options. Selenium automatically records each step as part of the test case. After completing all intended actions for the test, we simply click "Stop" to finish the recording.

3. Description of the test cases developed:

Each page included Home, Browse, Favorites, and Contact were tested separately to ensure the presence and functionality of the required GUI elements, including images, buttons, text boxes, labels, and additional components like checkboxes, sliders, and radio buttons. After individually verifying each page, a separate test case was created to validate the overall flow between the pages, making sure the navigation sequence worked as intended across the entire application.

a. Homepage/index_page test case

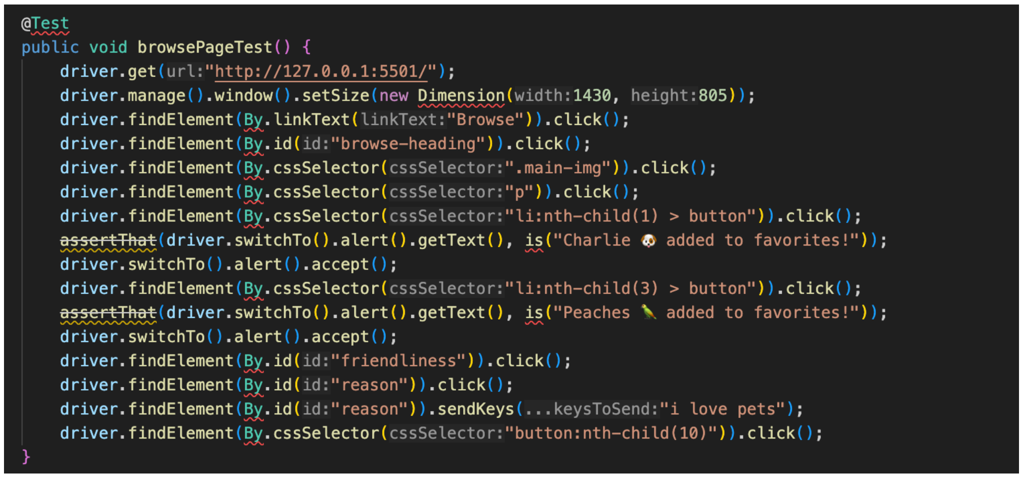
This test case verifies the presence and functionality of GUI elements on the homepage, including an image, a newsletter input textbox, radio buttons for pet selection, and navigation buttons. It simulates a user filling out the newsletter form and making selections, ensuring all required GUI components are functional according to the project guidelines.



```java
@Test
public void homepageTest() {
    driver.get(url:"http://127.0.0.1:5501/");
    driver.manage().window().setSize(new Dimension(width:1430, height:805));
    driver.findElement(By.linkText(linkText:"Home")).click();
    driver.findElement(By.id(id:"home-heading")).click();
    driver.findElement(By.cssSelector(cssSelector:".main-img")).click();
    driver.findElement(By.id(id:"newsletter")).click();
    driver.findElement(By.id(id:"newsletter")).sendKeys(...keysToSend:"test@example.com");
    driver.findElement(By.id(id:"agree")).click();
    driver.findElement(By.cssSelector(cssSelector:"label:nth-child(2) > input")).click();
    driver.findElement(By.cssSelector(cssSelector:"label:nth-child(3) > input")).click();
    driver.findElement(By.cssSelector(cssSelector:"button")).click();
}
```

b. Browse_page test case

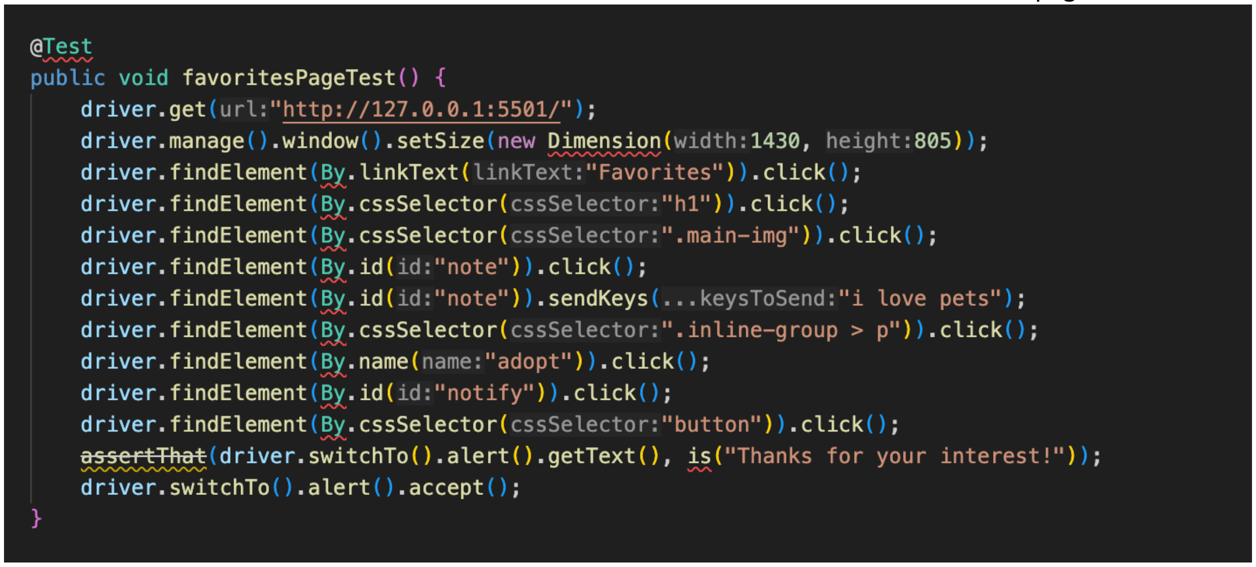
The browse page test confirms that users can view available pets, add favorites, and interact with elements such as the friendliness slider and text area for adoption reasons. The test validates the list of pets and captures alerts when pets are favorited, ensuring dynamic interaction is working as expected.



```java
@Test
public void browsePageTest() {
    driver.get(url:"http://127.0.0.1:5501/");
    driver.manage().window().setSize(new Dimension(width:1430, height:805));
    driver.findElement(By.linkText(linkText:"Browse")).click();
    driver.findElement(By.id(id:"browse-heading")).click();
    driver.findElement(By.cssSelector(cssSelector:".main-img")).click();
    driver.findElement(By.cssSelector(cssSelector:"p")).click();
    driver.findElement(By.cssSelector(cssSelector:"li:nth-child(1) > button")).click();
    ~~assertThat~~(driver.switchTo().alert().getText(), is("Charlie 🐶 added to favorites!"));
    driver.switchTo().alert().accept();
    driver.findElement(By.cssSelector(cssSelector:"li:nth-child(3) > button")).click();
    ~~assertThat~~(driver.switchTo().alert().getText(), is("Peaches 🦜 added to favorites!"));
    driver.switchTo().alert().accept();
    driver.findElement(By.id(id:"friendliness")).click();
    driver.findElement(By.id(id:"reason")).click();
    driver.findElement(By.id(id:"reason")).sendKeys(...keysToSend:"i love pets");
    driver.findElement(By.cssSelector(cssSelector:"button:nth-child(10)")).click();
}
```

c. Favorites page test case

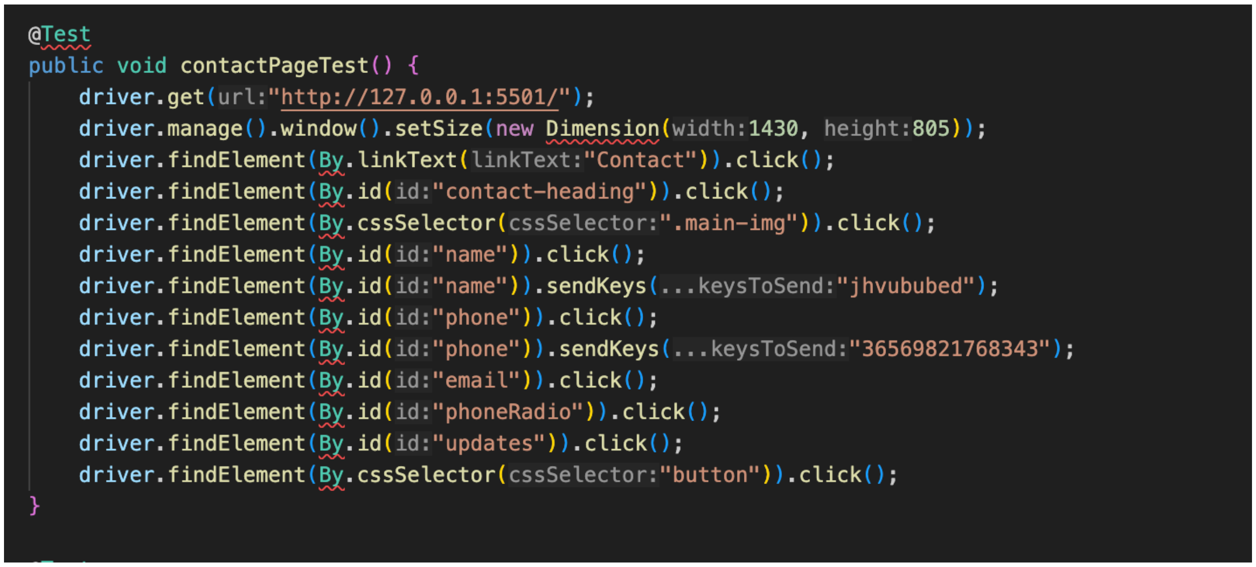
This test checks that selected favorite pets are properly displayed in the favorites list. It also verifies that users can rate their favorite pets using a slider and interact with the clear favorites functionality. The test ensures correct element behavior after users have made selections in the browse page.



```java
@Test
public void favoritesPageTest() {
    driver.get(url:"http://127.0.0.1:5501/");
    driver.manage().window().setSize(new Dimension(width:1430, height:805));
    driver.findElement(By.linkText(linkText:"Favorites")).click();
    driver.findElement(By.cssSelector(cssSelector:"h1")).click();
    driver.findElement(By.cssSelector(cssSelector:".main-img")).click();
    driver.findElement(By.id(id:"note")).click();
    driver.findElement(By.id(id:"note")).sendKeys(...keysToSend:"i love pets");
    driver.findElement(By.cssSelector(cssSelector:".inline-group > p")).click();
    driver.findElement(By.name(name:"adopt")).click();
    driver.findElement(By.id(id:"notify")).click();
    driver.findElement(By.cssSelector(cssSelector:"button")).click();
    ~~assertThat~~(driver.switchTo().alert().getText(), is("Thanks for your interest!"));
    driver.switchTo().alert().accept();
}
```

d. Contact page test case

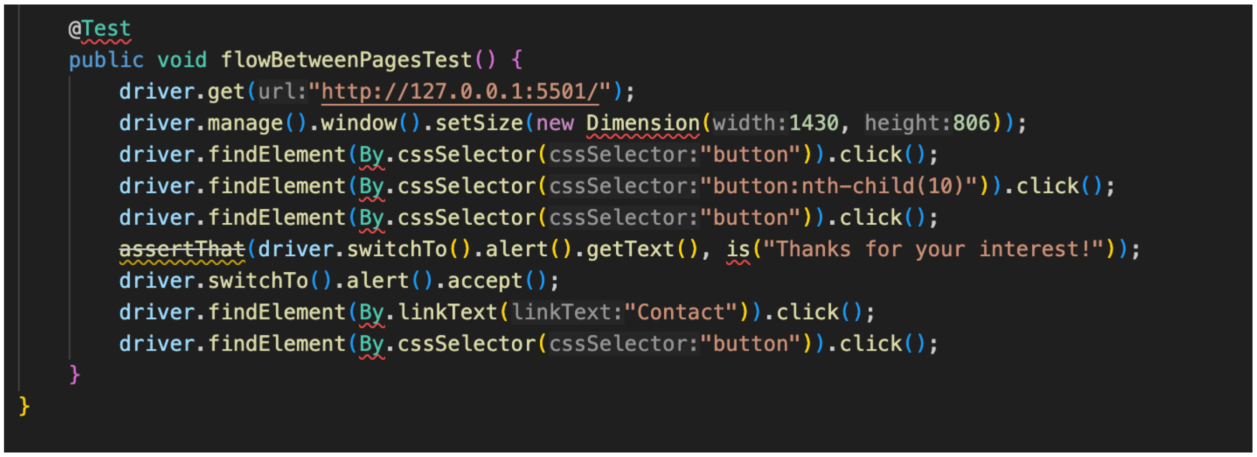
The contact page test simulates completing the contact form, including filling in the name and phone fields, selecting a preferred method of contact using radio buttons, and opting in for updates via a checkbox. It ensures form submission interactions are captured accurately and required fields are functional.



```java
@Test
public void contactPageTest() {
    driver.get(url:"http://127.0.0.1:5501/");
    driver.manage().window().setSize(new Dimension(width:1430, height:805));
    driver.findElement(By.linkText(linkText:"Contact")).click();
    driver.findElement(By.id(id:"contact-heading")).click();
    driver.findElement(By.cssSelector(cssSelector:".main-img")).click();
    driver.findElement(By.id(id:"name")).click();
    driver.findElement(By.id(id:"name")).sendKeys(...keysToSend:"jhvububed");
    driver.findElement(By.id(id:"phone")).click();
    driver.findElement(By.id(id:"phone")).sendKeys(...keysToSend:"36569821768343");
    driver.findElement(By.id(id:"email")).click();
    driver.findElement(By.id(id:"phoneRadio")).click();
    driver.findElement(By.id(id:"updates")).click();
    driver.findElement(By.cssSelector(cssSelector:"button")).click();
}
```

e. Flow between pages test case

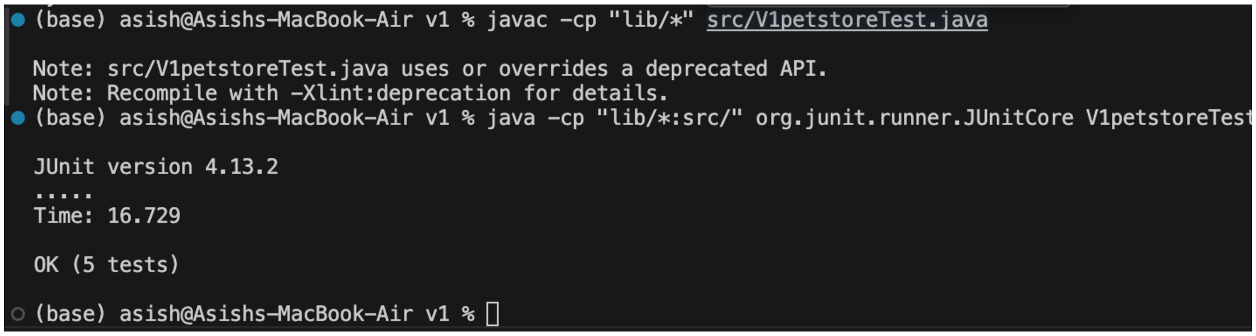
This test validates the navigation flow across the homepage, browse page, favorites page, and contact page. It simulates clicking buttons and moving through the application to ensure the page transitions are smooth and that the flow requirements of the assignment are fully met.


```
    @Test
    public void flowBetweenPagesTest() {
        driver.get(url:"http://127.0.0.1:5501/");
        driver.manage().window().setSize(new Dimension(width:1430, height:806));
        driver.findElement(By.cssSelector(cssSelector:"button")).click();
        driver.findElement(By.cssSelector(cssSelector:"button:nth-child(10)")).click();
        driver.findElement(By.cssSelector(cssSelector:"button")).click();
        ~~assertThat~~(driver.switchTo().alert().getText(), is("Thanks for your interest!"));
        driver.switchTo().alert().accept();
        driver.findElement(By.linkText(linkText:"Contact")).click();
        driver.findElement(By.cssSelector(cssSelector:"button")).click();
    }
}
```

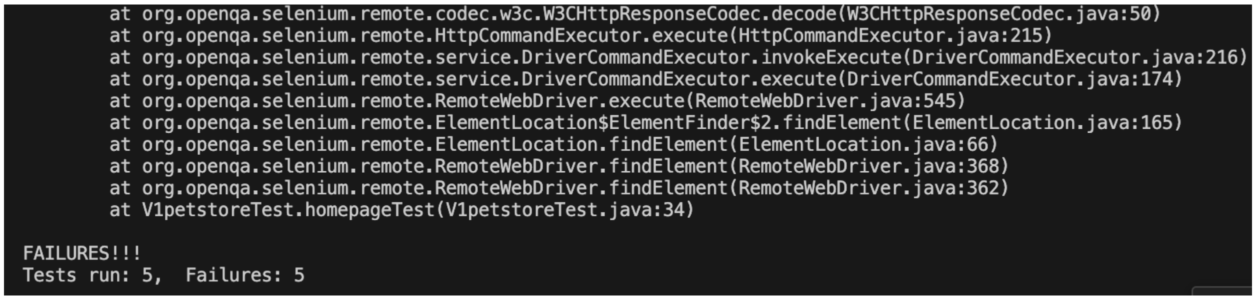
Overall, For GUI automation testing, I used Selenium WebDriver with Java and structured the test cases using JUnit. To compile and run the Selenium tests, I used the javac command with the Selenium libraries in the classpath, and executed the tests using the JUnitCore runner. The application was hosted locally using a simple HTTP server, and the tests were run against the local server URL.

4. Explanation of the test results:

Test Results for Version 1 of the Developed GUI Application


```
(base) asish@Asishs-MacBook-Air v1 % javac -cp "lib/*" src/V1petstoreTest.java

Note: src/V1petstoreTest.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
(base) asish@Asishs-MacBook-Air v1 % java -cp "lib/*:src/" org.junit.runner.JUnitCore V1petstoreTest

JUnit version 4.13.2
.....
Time: 16.729

OK (5 tests)

(base) asish@Asishs-MacBook-Air v1 %
```

Test Results for Version 2 of the Developed GUI Application

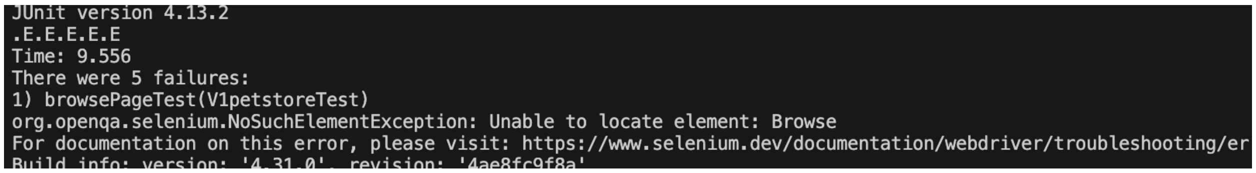

```
        at org.openqa.selenium.remote.codec.w3c.W3CHttpResponseCodec.decode(W3CHttpResponseCodec.java:50)
        at org.openqa.selenium.remote.HttpCommandExecutor.execute(HttpCommandExecutor.java:215)
        at org.openqa.selenium.remote.service.DriverCommandExecutor.invokeExecute(DriverCommandExecutor.java:216)
        at org.openqa.selenium.remote.service.DriverCommandExecutor.execute(DriverCommandExecutor.java:174)
        at org.openqa.selenium.remote.RemoteWebDriver.execute(RemoteWebDriver.java:545)
        at org.openqa.selenium.remote.ElementLocation$ElementFinder$2.findElement(ElementLocation.java:165)
        at org.openqa.selenium.remote.ElementLocation.findElement(ElementLocation.java:66)
        at org.openqa.selenium.remote.RemoteWebDriver.findElement(RemoteWebDriver.java:368)
        at org.openqa.selenium.remote.RemoteWebDriver.findElement(RemoteWebDriver.java:362)
        at V1petstoreTest.homepageTest(V1petstoreTest.java:34)

FAILURES!!!
Tests run: 5,  Failures: 5
```

As we can see that the mentioned test cases have failed. The reason for this is that the test case failures occurred primarily due to intentional GUI modifications introduced in Version 2, such as reordering navigation links, altering button placements, and removing or changing element identifiers. These failures were expected as part of the experiment to evaluate the sensitivity of Selenium scripts to GUI changes. And I have made sure that I have clicked certain elements in the version 1 to make sure that the label and the position is recorded in the version1 of the test case like mentioned.

I made sure that each single test in the test cases developed would fail and here are the reasons with the screenshots.

1. browsePageTest Failure — #browse-heading Not Found

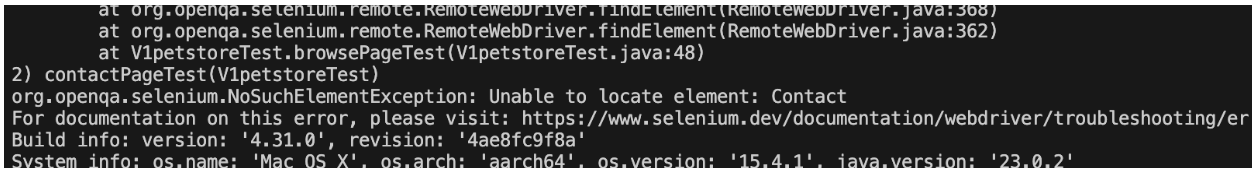
- **Change Introduced:** In Version 2, the browse page's main heading (id="browse-heading") was removed or replaced with a styled <h1> without an ID.
- **Impact on Test Case:** The Selenium test was written to locate the heading by its ID. Since the ID no longer exists in Version 2, the test failed with a "NoSuchElementException."



```
JUnit version 4.13.2
.E.E.E.E.E
Time: 9.556
There were 5 failures:
1) browsePageTest(V1petstoreTest)
org.openqa.selenium.NoSuchElementException: Unable to locate element: Browse
For documentation on this error, please visit: https://www.selenium.dev/documentation/webdriver/troubleshooting/er
Build info: version: '4.31.0', revision: '4ae8fc9f8a'
```

2. contactPageTest Failure — Contact Link Not Found

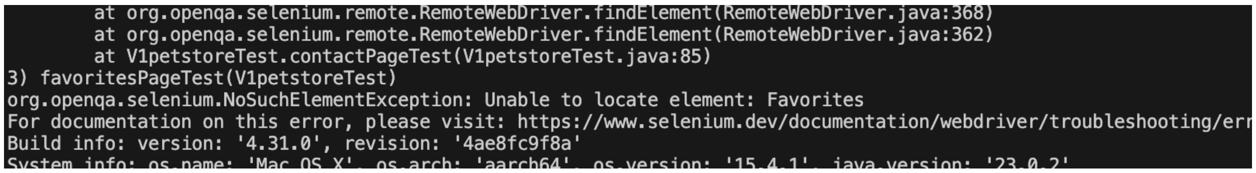
- **Change Introduced:** In Version 2, the navigation bar layout and link structure were modified — links were reordered, and styling was changed.
- **Impact on Test Case:** The Selenium test expected to find the "Contact" page using the old link text lookup. With the new navigation structure, the locator could no longer find "Contact," leading to a failure.



```
        at org.openqa.selenium.remote.RemoteWebDriver.findElement(RemoteWebDriver.java:368)
        at org.openqa.selenium.remote.RemoteWebDriver.findElement(RemoteWebDriver.java:362)
        at V1petstoreTest.browsePageTest(V1petstoreTest.java:48)
2) contactPageTest(V1petstoreTest)
org.openqa.selenium.NoSuchElementException: Unable to locate element: Contact
For documentation on this error, please visit: https://www.selenium.dev/documentation/webdriver/troubleshooting/er
Build info: version: '4.31.0', revision: '4ae8fc9f8a'
System info: os.name: 'Mac OS X', os.arch: 'aarch64', os.version: '15.4.1', java.version: '23.0.2'
```

3. favoritesPageTest Failure — Favorites Link Not Found

- **Change Introduced:** The Favorites page navigation link was moved in Version 2. Also, the way pets were displayed and interacted with was slightly redesigned.
- **Impact on Test Case:** Selenium tried to find "Favorites" by direct link text, but due to the new structure, it couldn't locate it — causing the test case to fail.

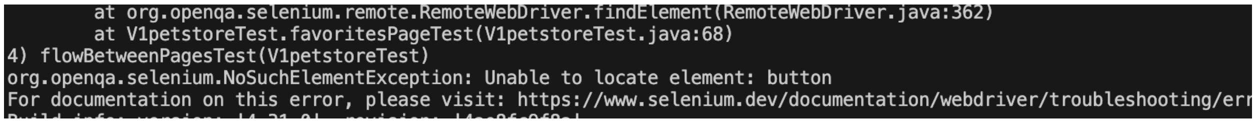


```
        at org.openqa.selenium.remote.RemoteWebDriver.findElement(RemoteWebDriver.java:368)
        at org.openqa.selenium.remote.RemoteWebDriver.findElement(RemoteWebDriver.java:362)
        at V1petstoreTest.contactPageTest(V1petstoreTest.java:85)
3) favoritesPageTest(V1petstoreTest)
org.openqa.selenium.NoSuchElementException: Unable to locate element: Favorites
For documentation on this error, please visit: https://www.selenium.dev/documentation/webdriver/troubleshooting/err
Build info: version: '4.31.0', revision: '4ae8fc9f8a'
System info: os.name: 'Mac OS X', os.arch: 'aarch64', os.version: '15.4.1', java.version: '23.0.2'
```

flowBetweenPagesTest Failure — button Not Found

- **Change Introduced:** Page flow was intentionally changed in Version 2: buttons were moved and reorderedfor navigating between pages.

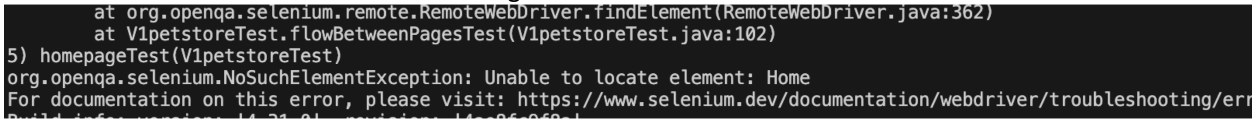
- **Impact on Test Case:** The test originally relied on the specific sequence and position of buttons (like nth-child(10)). Since the button order changed, Selenium couldn't find the expected button and failed.



```
        at org.openqa.selenium.remote.RemoteWebDriver.findElement(RemoteWebDriver.java:362)
        at V1petstoreTest.favoritesPageTest(V1petstoreTest.java:68)
4) flowBetweenPagesTest(V1petstoreTest)
org.openqa.selenium.NoSuchElementException: Unable to locate element: button
For documentation on this error, please visit: https://www.selenium.dev/documentation/webdriver/troubleshooting/err
```

5. homepageTest Failure — Home Link Not Found

- **Change Introduced:** On the homepage, the navigation link for "Home" was reordered and/or the link text structure was updated.
- **Impact on Test Case:** The Selenium script, which depended on finding "Home" via link text, couldn't detect the new structure — resulting in test failure.



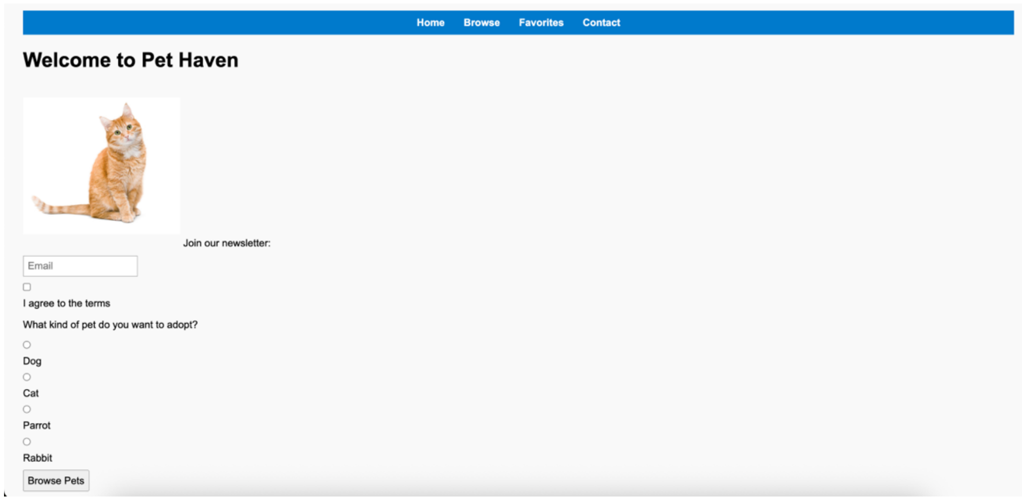
```
        at org.openqa.selenium.remote.RemoteWebDriver.findElement(RemoteWebDriver.java:362)
        at V1petstoreTest.flowBetweenPagesTest(V1petstoreTest.java:102)
5) homepageTest(V1petstoreTest)
org.openqa.selenium.NoSuchElementException: Unable to locate element: Home
For documentation on this error, please visit: https://www.selenium.dev/documentation/webdriver/troubleshooting/err
```

Changes done when compared to version1 and version 2:

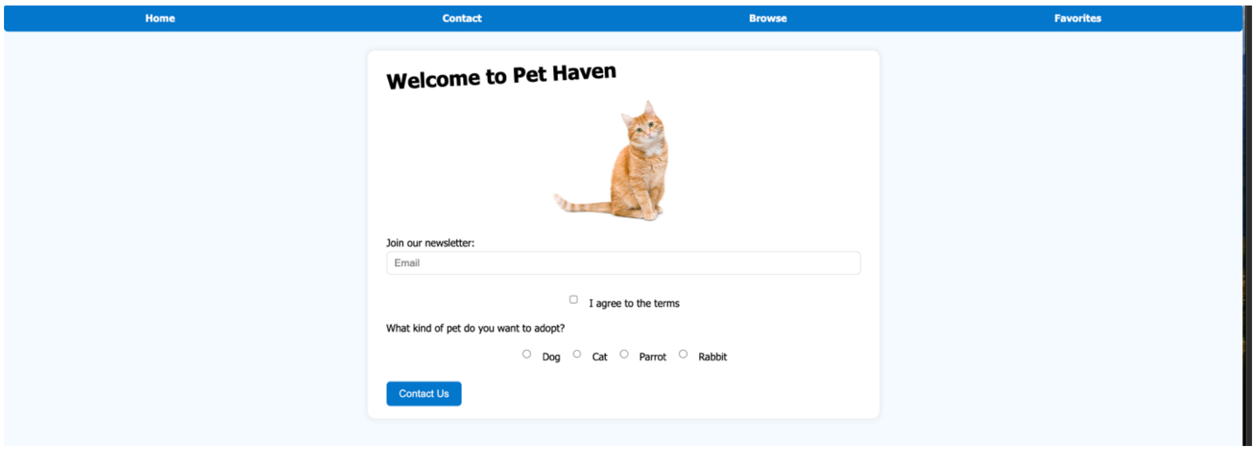
Homepage Changes

In Version 2, the homepage heading styling was changed (rotated slightly for visual effect), and the layout was centered more prominently. The form elements, such as the email input and pet selection radio buttons, were visually reorganized into tighter groups. Additionally, button placements were changed to create a more mobile-friendly and cleaner look.

Version 1:



Version 2:



Welcome to Pet Haven

Join our newsletter:

Email

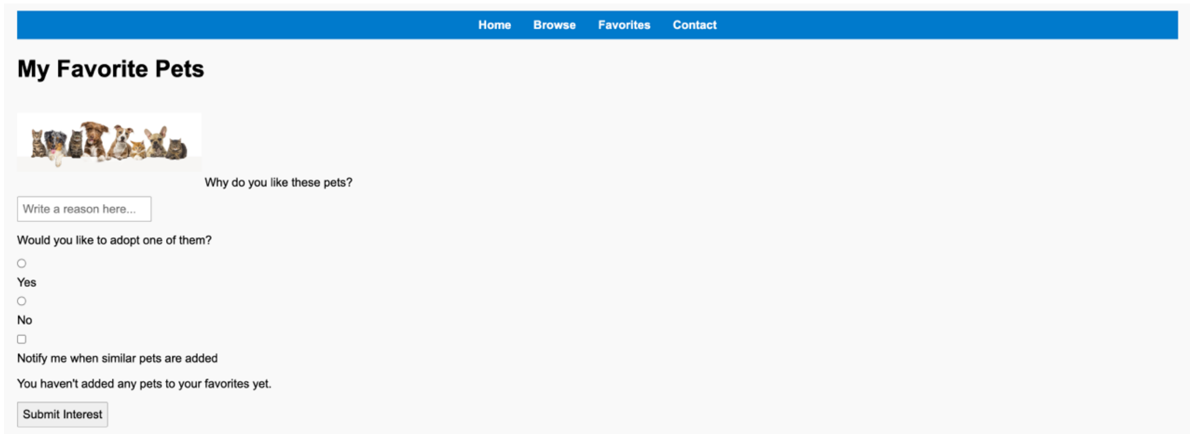
I agree to the terms

What kind of pet do you want to adopt?

Dog Cat Parrot Rabbit

Contact Us

Favorites Page Changes
The Favorites page in Version 2 introduced a cleaner, minimal layout. The pet list was dynamically populated, and additional UI elements like a rating slider and a "Clear Favorites" button were updated in style and position. Some elements like images and multi-line notes seen in Version 1 were removed to streamline the page.
Version 1:



My Favorite Pets

Why do you like these pets?

Write a reason here...

Would you like to adopt one of them?

Yes

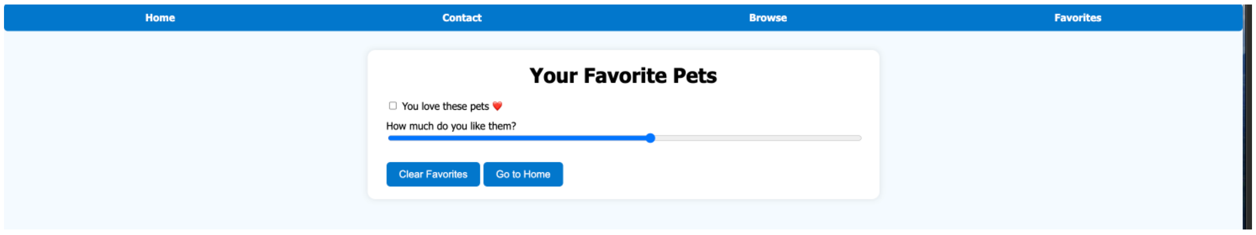
No

Notify me when similar pets are added

You haven't added any pets to your favorites yet.

Submit Interest

Version 2:



Your Favorite Pets

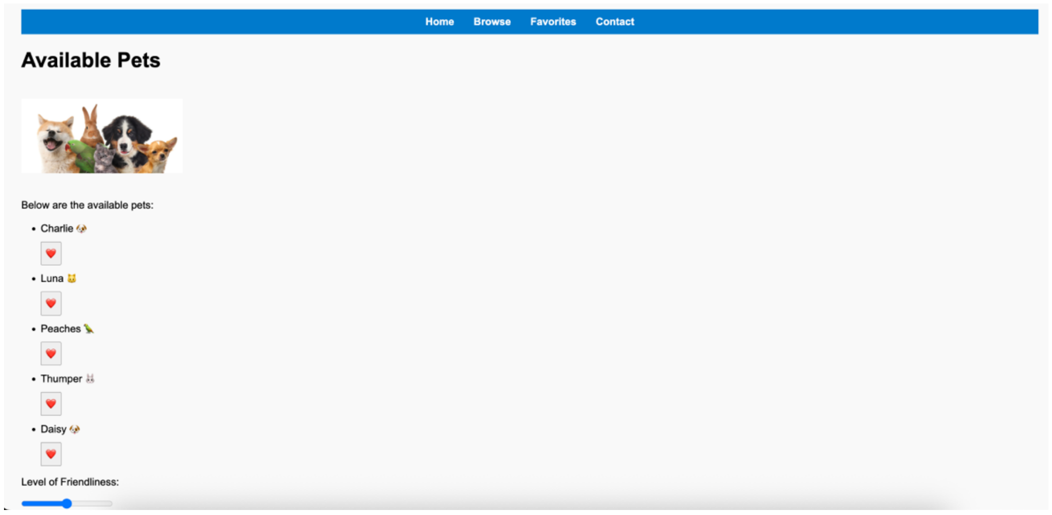
You love these pets ❤️

How much do you like them?

Clear Favorites Go to Home

Browse Page Changes
On the Browse page, the pet listing layout was simplified in Version 2. The interactive buttons for adding favorites were moved inside the list items, and stylistic changes like enlarged headings and a centered "Choose your favorite pets" prompt were introduced. Additionally, the "Go to Favorites" navigation was repositioned for better flow.

Version 1:



Home Browse Favorites Contact

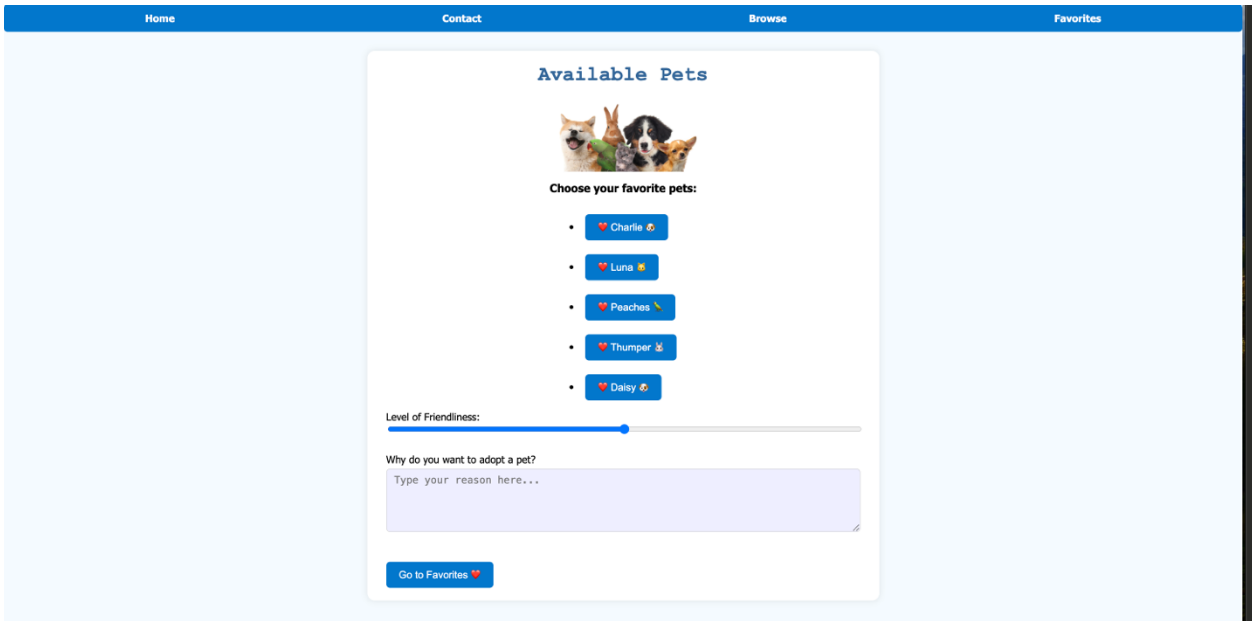
Available Pets

Below are the available pets:

- Charlie 🐶
  ❤️
- Luna 🐱
  ❤️
- Peaches 🦜
  ❤️
- Thumper 🐰
  ❤️
- Daisy 🐶
  ❤️

Level of Friendliness:

Version 2:



Home Contact Browse Favorites

Available Pets

Choose your favorite pets:

- ❤️ Charlie 🐶
- ❤️ Luna 🐱
- ❤️ Peaches 🦜
- ❤️ Thumper 🐰
- ❤️ Daisy 🐶

Level of Friendliness:

Why do you want to adopt a pet?

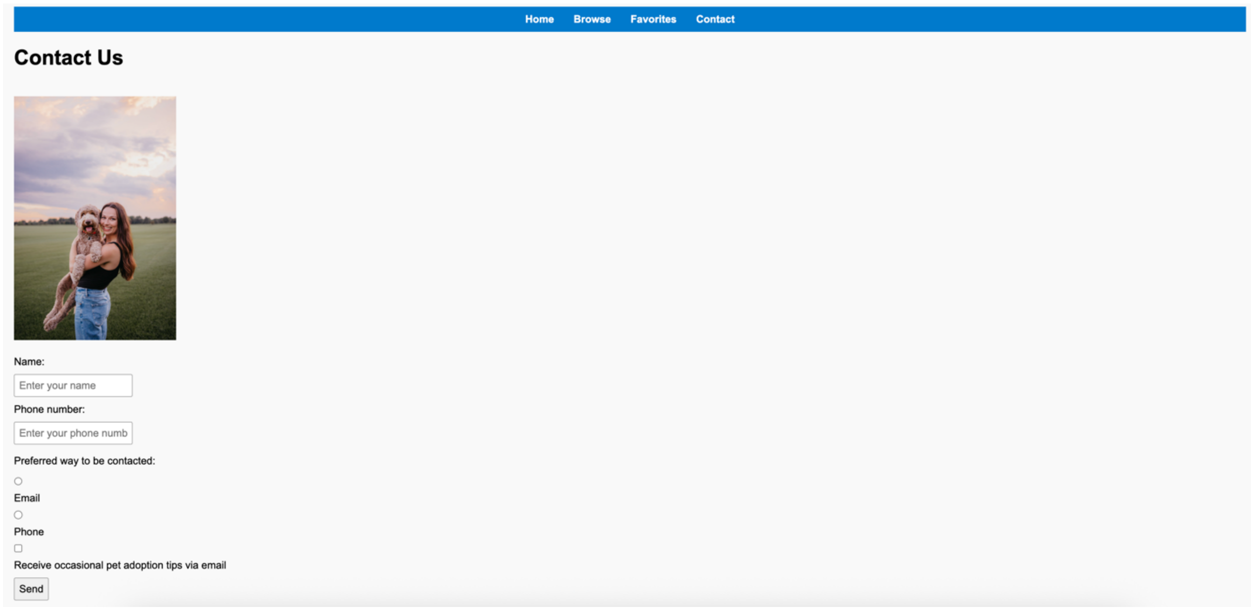
Type your reason here...

Go to Favorites ❤️

Contact Page Changes

The Contact page was visually redesigned with a new font style for the heading ("Courier New" monospace), a more centered and compact form layout, and reorganized checkboxes and radio buttons for easier selection. The browse navigation button was added directly inside the form for better usability.

Version1:



Contact Us

Name:

Enter your name

Phone number:

Enter your phone numb

Preferred way to be contacted:

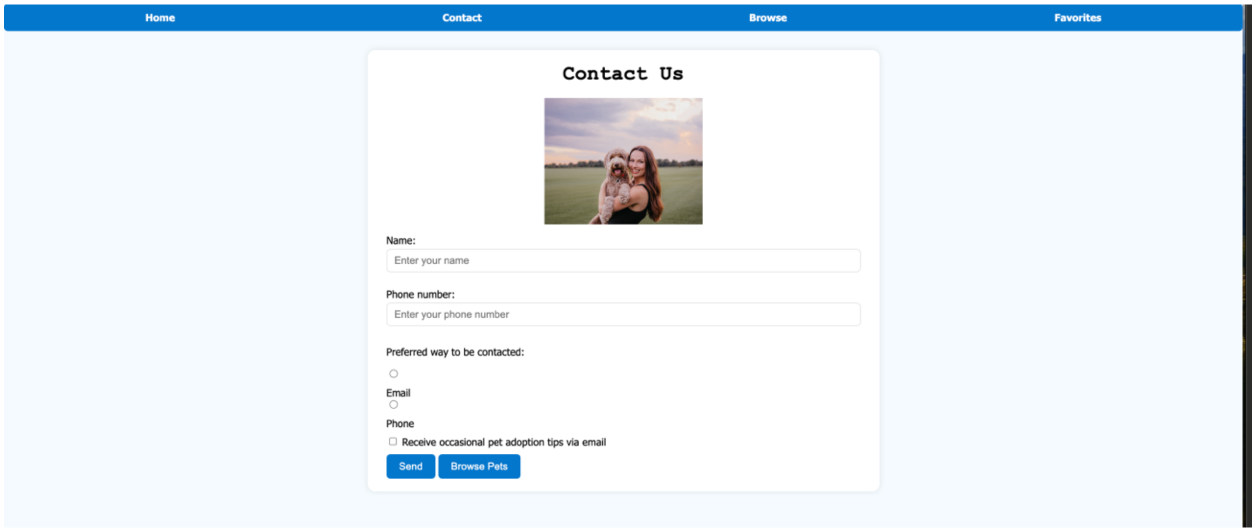
Email

Phone

Receive occasional pet adoption tips via email

Send

Version2:



Contact Us

Name:

Enter your name

Phone number:

Enter your phone number

Preferred way to be contacted:

Email

Phone

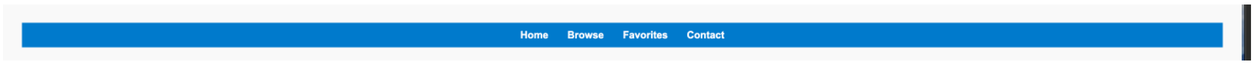
Receive occasional pet adoption tips via email

Send Browse Pets

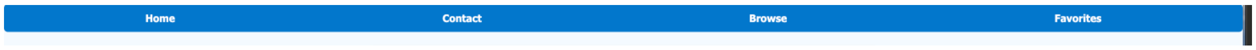
Overall Navigation and Flow Changes

Across all pages, the navigation bar was reordered: "Home" and "Contact" links were swapped in sequence compared to Version 1. Button navigation between pages (especially in Favorites and Contact pages) was adjusted to create a new flow. Some direct navigation links that existed in Version 1 were altered or removed, forcing users to follow a slightly different path across pages in Version 2.

Version 1:



Version2:



5. Assesment of the GUI tool:

Set of Features and Functionalities Provided
Selenium WebDriver offers a comprehensive set of features for GUI automation. It allows simulation of real user actions such as clicking, typing, scrolling, selecting options, and navigating across multiple pages. It supports assertions for verifying text, size, location, and existence of GUI elements. Additionally, Selenium provides flexible element locators (by ID, name, link text, CSS selector, XPath), advanced actions (drag-and-drop, hover), and integration with testing frameworks like JUnit, making it highly adaptable for complex testing needs.

Type of Coverage
The type of testing coverage Selenium enabled was quite broad. It handled testing individual elements (like verifying the presence and size of labels, sliders, and buttons) as well as complete user flows (moving between pages, submitting forms, handling alerts). The tool worked across different pages of the application, making sure each page's functionality and layout were correctly verified against expectations.

Reuse of Test Cases
Selenium made it relatively easy to reuse the test cases between Version 1 and Version 2, especially because each page had its own modular test. However, when the GUI structure changed (like element IDs being removed or button positions shifting), some adjustments were needed in the locators. Still, the overall framework of tests remained reusable, saving a lot of effort compared to starting from scratch.

Test Results Produced
The tool produced clear, immediate feedback on test execution. Each test either passed or failed explicitly, with detailed error messages when a failure occurred (such as "NoSuchElementException" when an element was not found). This made debugging efficient because it was easy to trace whether a test failed because of a missing element, a wrong flow, or a locator mismatch.

Ease of Usage
While Selenium was powerful, it required some initial setup work — downloading the correct drivers (like geckodriver for Firefox), configuring them, and making sure Java libraries were correctly linked. Learning how to find stable locators and deal with timing issues (for dynamic elements) took some practice. But once set up, writing and modifying the test cases became quite smooth, and the scripts were easy to maintain.

Type of GUI Elements That Can Be Tested
Selenium has the ability to test a wide range of GUI elements in the project. These included:

- Text fields and text areas
- Buttons and clickable elements
- Radio buttons and checkboxes
- Sliders (input type range)
- Navigation links
- Images and labels
- Pop-up alerts This made Selenium a good fit for the type of web application developed for this assignment.

Challenges and Limitations Observed
Although Selenium WebDriver was powerful, a few challenges were noticeable during the project. The tests were quite sensitive to changes in the GUI — small modifications like updating an ID or repositioning a button

caused failures that required script adjustments. Handling dynamic content sometimes needed explicit waits to ensure elements were available before interacting with them. Additionally, the initial setup of drivers and environment configuration took some effort. Lastly, while the tool produced clear results, it lacked advanced visual reporting unless extra libraries were integrated.

6. References:

1. Selenium Project. (n.d.). *Selenium WebDriver*. Retrieved from https://www.selenium.dev/documentation/webdriver/

2. SeleniumHQ. (n.d.). *Selenium Overview: History, Components, and Usage*. Retrieved from https://www.selenium.dev/