Automation Testing and Final Reporting

Testing website :https://the-internet.herokuapp.com/login

Table of contents

1. Project Overview

The objective of this project was to perform end-to-end Quality Assurance testing on a demo web application. The testing activities were carried out over three weeks and included manual testing, test planning with black-box techniques, and basic automation testing using Selenium.

The primary module tested was the Login functionality, along with a post-login feature (Logout).

Selenium automation was performed by converting manual login test cases into automated scripts. The scripts executed browser actions such as opening the login page, entering credentials, clicking the login button, and verifying the results automatically.

Two test cases were automated using Selenium with Python. The first test case validated the login functionality using valid credentials. The second test case validated the secure area feature by performing logout functionality after successful login.

2. Tools and technologies used

- Manual Testing
- Selenium Automation Tool
- Python Programming Language
- Google Chrome Browser

3. Testing Methodology

Week 1 – Manual Testing

In Week 1, manual testing was performed on the login page of a demo web application. Test cases were written and executed to validate login behavior using valid and invalid credentials. A test report and bug report were prepared and submitted.

- Total Test Cases Executed: 8

- Bugs Identified: 1

Week 2 - Advance testing

In Week 2, advanced testing concepts were applied to the same login module.

The following activities were performed:

- Test Plan creation
- Black Box Testing
- Boundary Value Analysis (conceptual)
- Cross-Browser Testing
- Retesting of previously reported defect

Cross-browser testing was performed using Google Chrome, Mozilla Firefox, and Microsoft Edge. Minor UI differences were observed, but core functionality remained consistent.

Week 3 – Automation Testing

In Week 3, Selenium automation testing was implemented using Python. Manual login test cases were converted into automated scripts.

Two test cases were automated:

- Login validation using valid credentials
- Feature testing of Secure Area by performing logout functionality

Explicit waits were used to handle synchronization issues and ensure stable execution.

4. Test Case Summary

Test Type	**Number of test cases**
Manual test cases	8
Automated test cases	2

5. Automation Test Cases Executed

Test Case 1: Login Validation

- Objective: Verify successful login using valid credentials
- Result: Pass

Test Case 2: Feature Test (Logout)

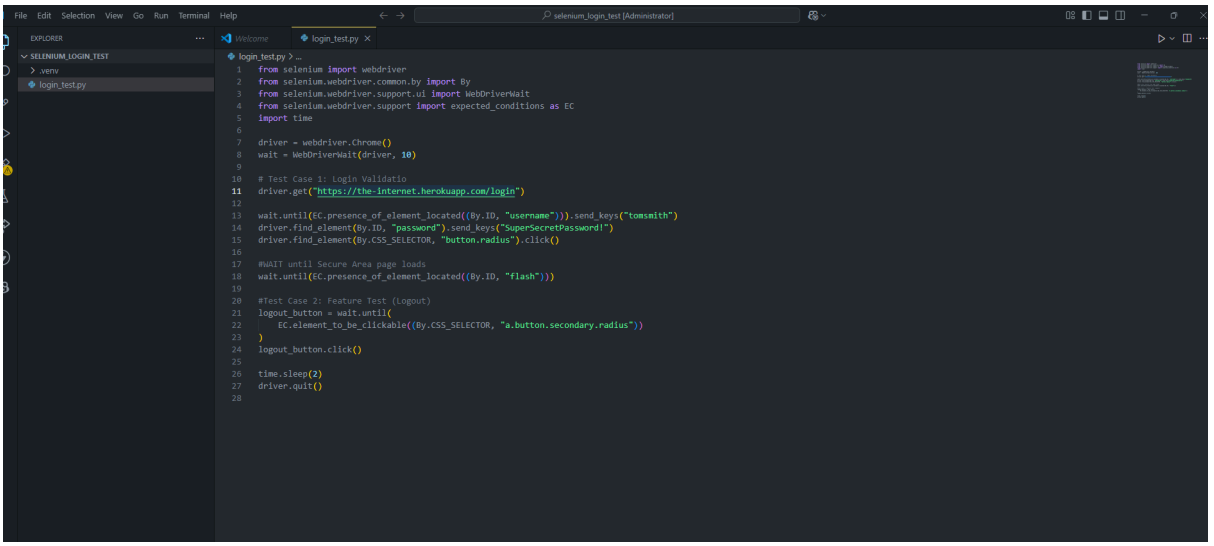
- Objective: Verify secure area access and logout functionality
- Result: Pass

6. Bug Report Summary

Bug ID	Description	Status
BUG_006	Incorrect error message displayed , when invalid username and valid password are entered	Open

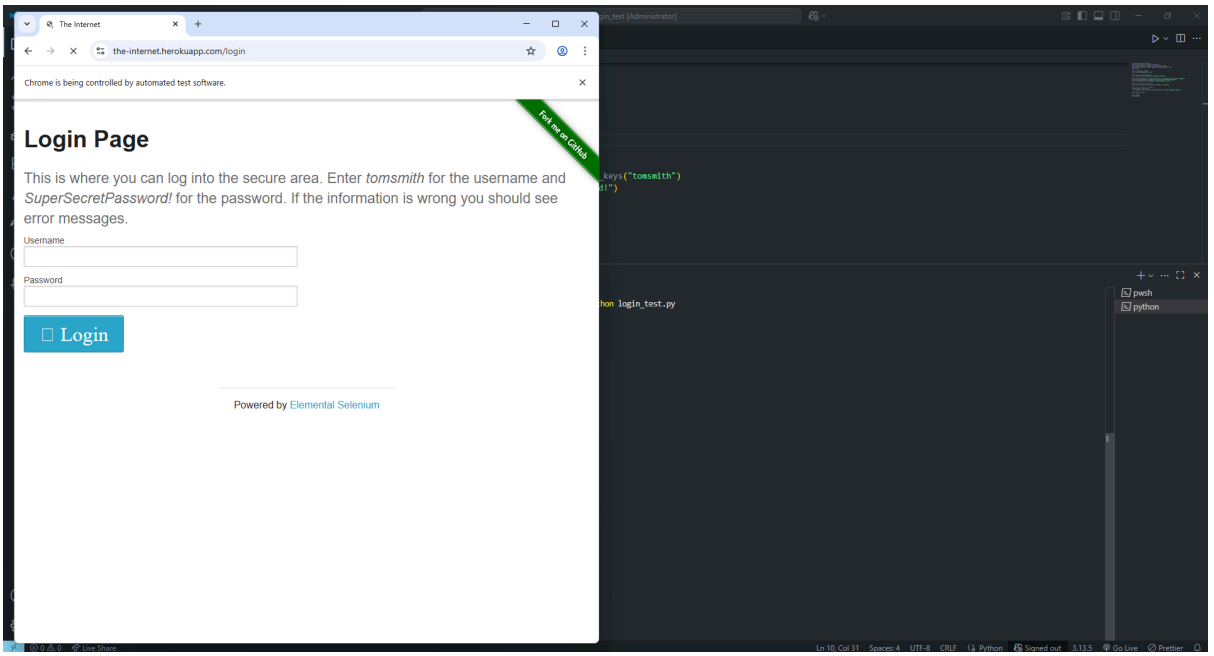
7. Automation Evidence

1. Selenium automation code

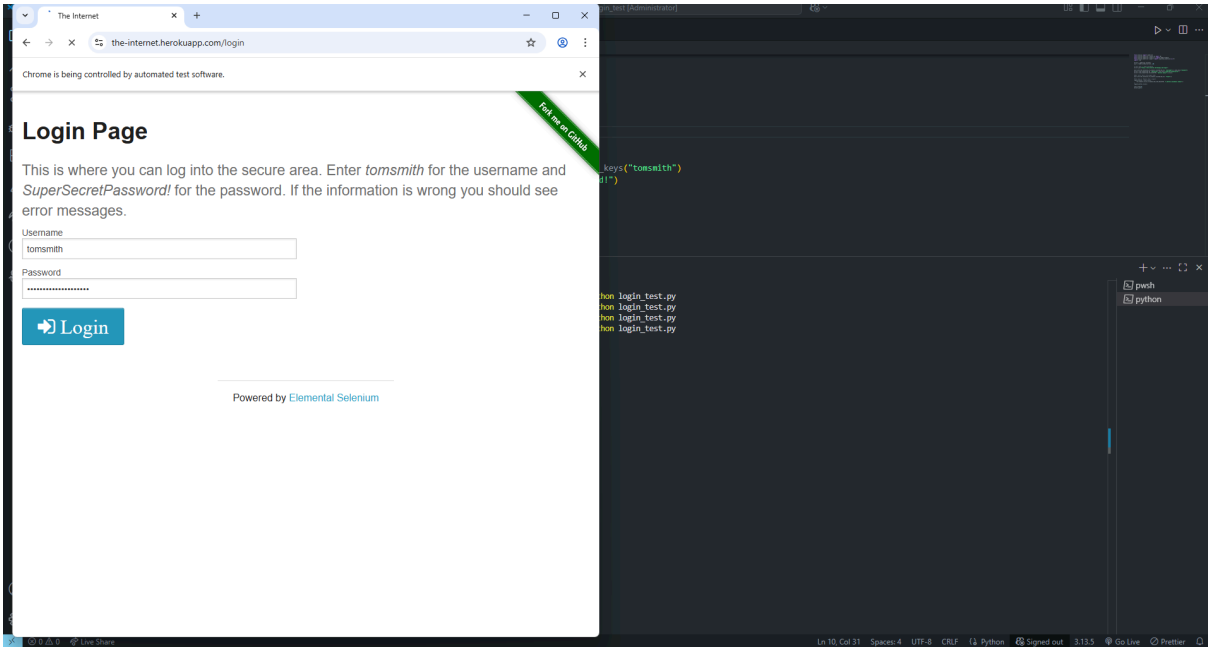


```python
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import time

driver = webdriver.Chrome()
wait = WebDriverWait(driver, 10)

# Test Case 1: Login Validatio
driver.get("https://the-internet.herokuapp.com/login")

wait.until(EC.presence_of_element_located((By.ID, "username"))).send_keys("tomsmith")
driver.find_element(By.ID, "password").send_keys("SuperSecretPassword!")
driver.find_element(By.CSS_SELECTOR, "button.radius").click()

#WAIT until Secure Area page loads
wait.until(EC.presence_of_element_located((By.ID, "flash")))

#Test Case 2: Feature Test (Logout)
logout_button = wait.until(
    EC.element_to_be_clickable((By.CSS_SELECTOR, "a.button.secondary.radius"))
)
logout_button.click()

time.sleep(2)
driver.quit()
```

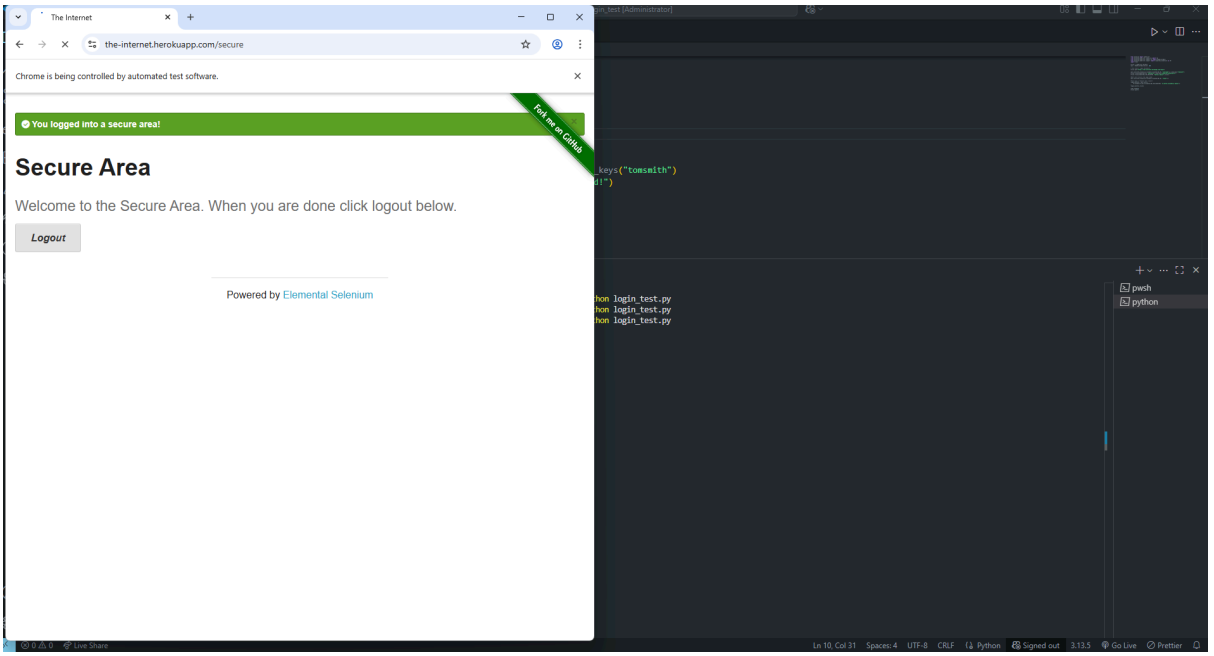
2. Automatic browser launch



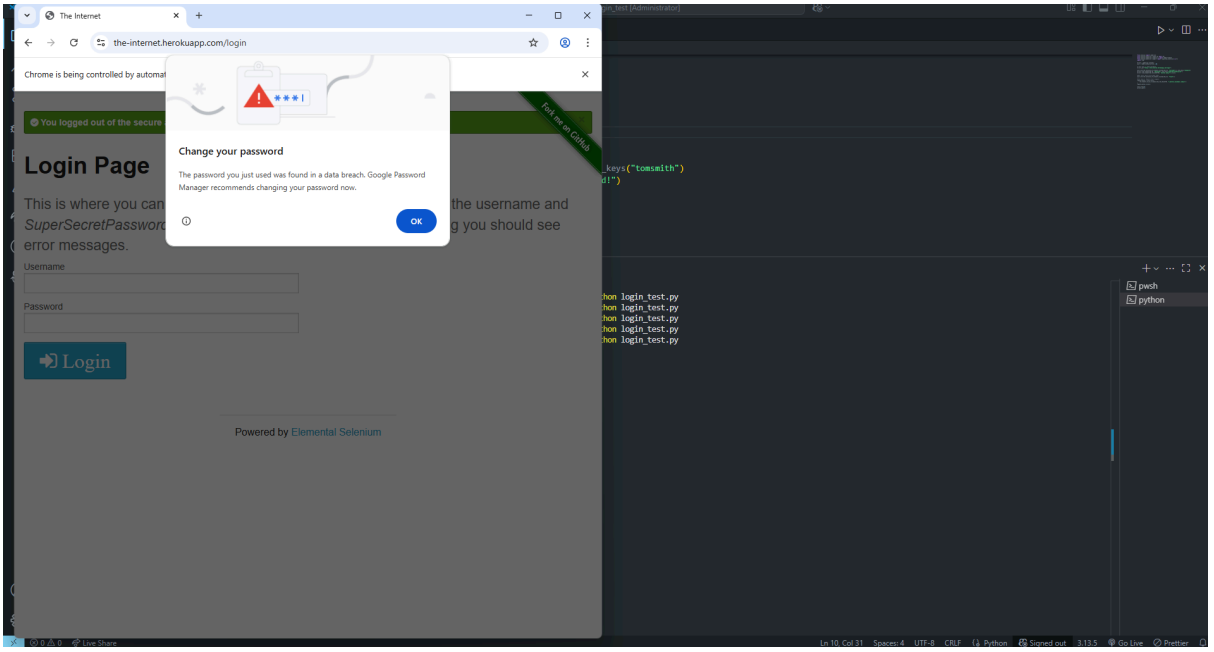
3. Credentials



4. Successful login page



5. Logout Functionality execution



8. Key Learnings

- Gained understanding of manual testing fundamentals
- Learned to create test plans and apply black-box testing techniques
- Understood the importance of cross-browser and retesting activities
- Learned basics of Selenium automation using Python
- Understood the difference between manual and automation testing

9. Conclusion

This project provided hands-on experience in software testing across manual, advanced, and automation levels. The testing activities ensured the login functionality worked correctly and consistently. Automation helped in validating repetitive test cases efficiently, while manual testing helped in identifying defects.