ROBOT Framework
Testing Project

Semester [CMP461] Team: APRIL

Authors

Evram Yousef
evramyousef@gmail.com
Sec. 1 Bn. 8

Omar Ahmed
omar.ahmed983@eng-st.cu.edu.eg
Sec. 1 Bn. 36

Kareem Osama
kareemosamasobeih@gmail.com
Sec. 2 Bn. 5

Muhammad Sayed
muhammad.mahmoud98@eng-st.cu.edu.eg
Sec. 2 Bn. 14

Supervised by: Dr. Ahmed Sobeih
Assisted by: Eng. Ali El-Seddeq

Table of Contents

Abstract

ROBOT Framework is a keyword-driven language that consists of many keywords to test user acceptance tests for websites. Throughout the ROBOT framework, we learned how to fully test websites, validate and verify their functionalities. The tool provided us with numerous useful techniques and libraries; to manage our testing processes efficiently. Using its comprehensive checkouts we managed to detect some flaws in our software under test (My Store website).

SUT

The software we are testing is a website under development called “My Store” (http://automationpractice.com/index.php), it’s an online e-commerce shopping website. That’s being used by the Selenium Framework website to help practice exercises on a real-time e-commerce website. The website consists of various components such as user profile, the home page, purchasing system, a navigation system, filtering techniques, and search component.

Applied Testing Techniques

We applied several testing techniques, listed as follows: First **Component** testing; specifying each independent page (or pages) that are providing one functionality as a Component, and provided its test cases individually, such as “Login and Logout”, “Sign up”, “Home Page”, “Profile”, “Search”, and “Navigation”. Secondly **Integration** testing; we took each related two or more components and tested them together, within the same test suites, such as “Sign up with Login and Logout”, “Sign up with Profile”, “Search with Home Page”, and so on. Then **System** testing, providing several use cases that involve the whole functionalities of the system, starting from Sign up, throughout Searching, Profile Updates, Purchasing, and finally logging out.

One last testing technique that we’ve applied is **Data-Driven Testing**, by applying various possible data in the SUT we’ve managed to detect serious issues with the website, unhandled situation, for example, a user may sign up with birth date (31 of February), usernames may have

spaces, and some other issues will be discussed at the demo. Also, we've detected infinite loops that occur in the website when certain data is applied to some component of the system.

Directories Specifications

In our deliveries, we appended three main folders, names "doc", "resources", and "test_suites".

"**doc**" is where our log files and reports are, ROBOT framework provides every test suite that's being called with a *report*, *log*, and an *output* XML file, to specify the characteristics, outputs, passed and failed test suites. This folder contains multiple folders, each one represents the logs for every component we've tested in the system.

"**resources**" is where we kept all the common keywords, and variables -*please notice that 'keywords' in the ROBOT framework is the alias for 'functions' in programming languages*- these keywords are called from different test suites (pages); that's why we've placed them in a separate place, grouping related keywords all together.

"***test_suites***", is the main part, test_suites contains multiple folders, each one of them holds the test cases of a component and/or multiple components. Each test case has a *tag* that specifies the type of the test (component, integrate, system, faulty). Component means that this component (eg. Sign up) is being tested separately, Integrate means that multiple components are involved, System means that this is a use case for the whole system testing, and Faulty means that this test case is directing a hazard that may cause a fault in the system, thus we are hoping that it would fail.

How To Use The Tool

0- If you don't have python or pip, please follow this link
https://phoenixnap.com/kb/install-pip-windows
1- Installation:
~ pip install robotframework
~ pip install robotframework-seleniumlibrary
~ pip install robotkernel
2- Download selenium web driver for chrome from this link
https://chromedriver.storage.googleapis.com/index.html?path=87.0.4280.88/
3- Usage:
To Run any subfile: change directory to the specific file and type
~ robot file_name.robot
OR to run System Test case type:
~ robot test_suites/system_test/system_test.robot
Or to run Data-Driven test cases type:
~ robot test_suites/system_test/data_driven.robot
Note: pages test suites (test cases/scenarios) are contained in the test_suites folder, with each page (or pages) asserting a functionality are grouped together.

ROBOT File Structure

A ROBOT file consists of four sections:

1. Settings: This section includes all used packages, resources, standard, and user-defined libraries.
2. Variables: contains global variables used in the file.
3. Keywords: it contains definitions of keywords each of them refers to sum code sentences.
4. Testcases: it contains some test cases each of which has its documentations and tags that define its purpose. The code of the test case is composed of some keywords defined in the keywords section or defined in the included libraries.

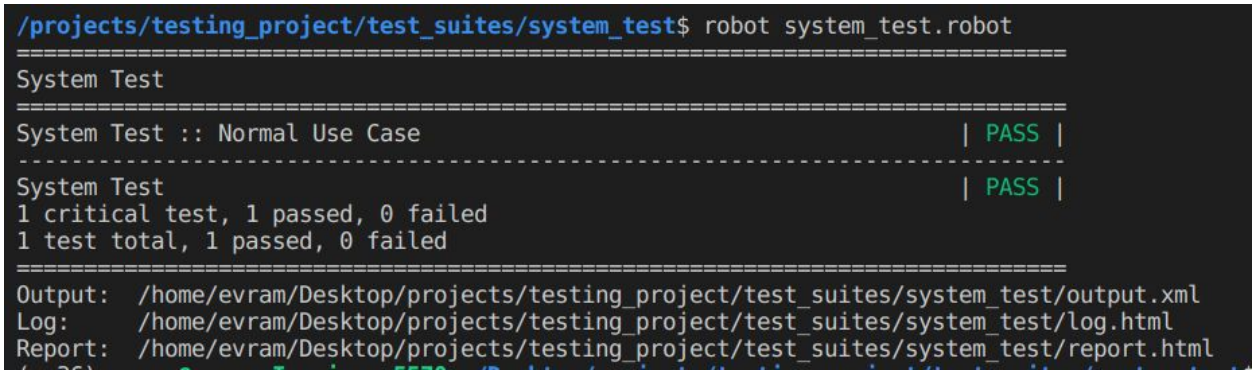
How To Run ROBOT File

- Open the terminal from the directory of the robot file and type: *~ robot file_name.robot*

Example:

Let's run the System Test case, which includes testing the system as a whole, throughout various functionalities that the system has.

~ robot test_suites/system_test/system_test.robot

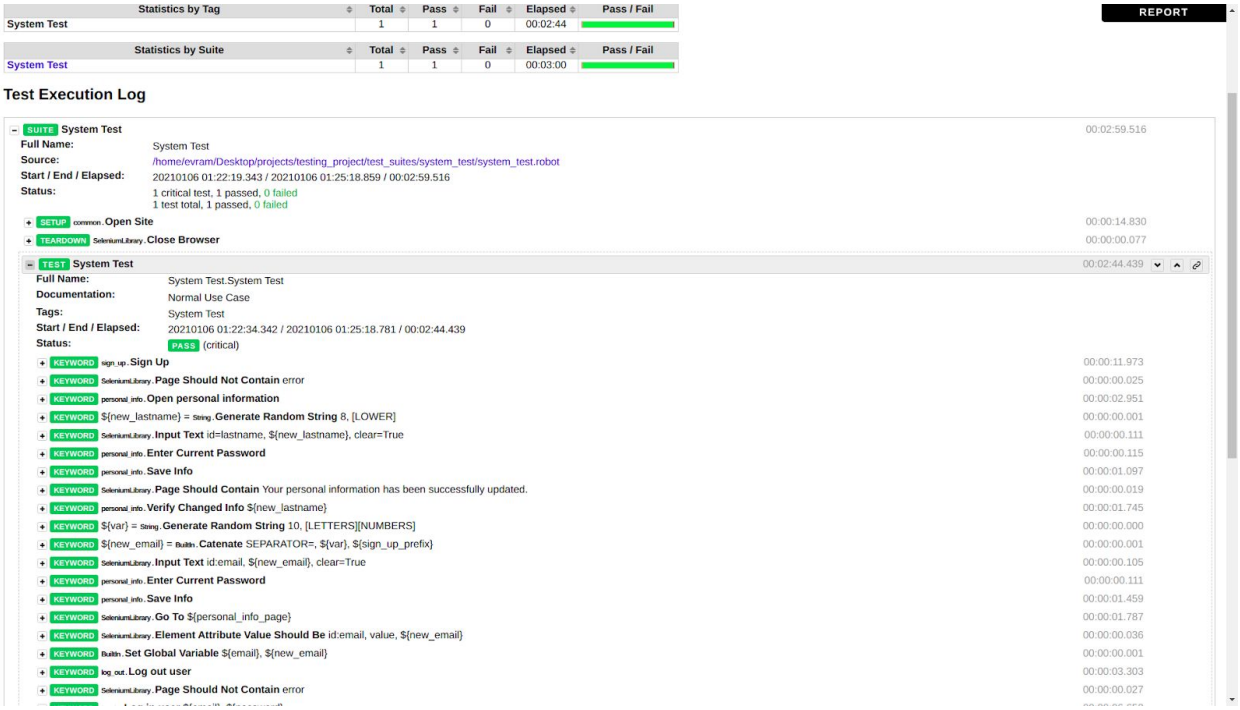


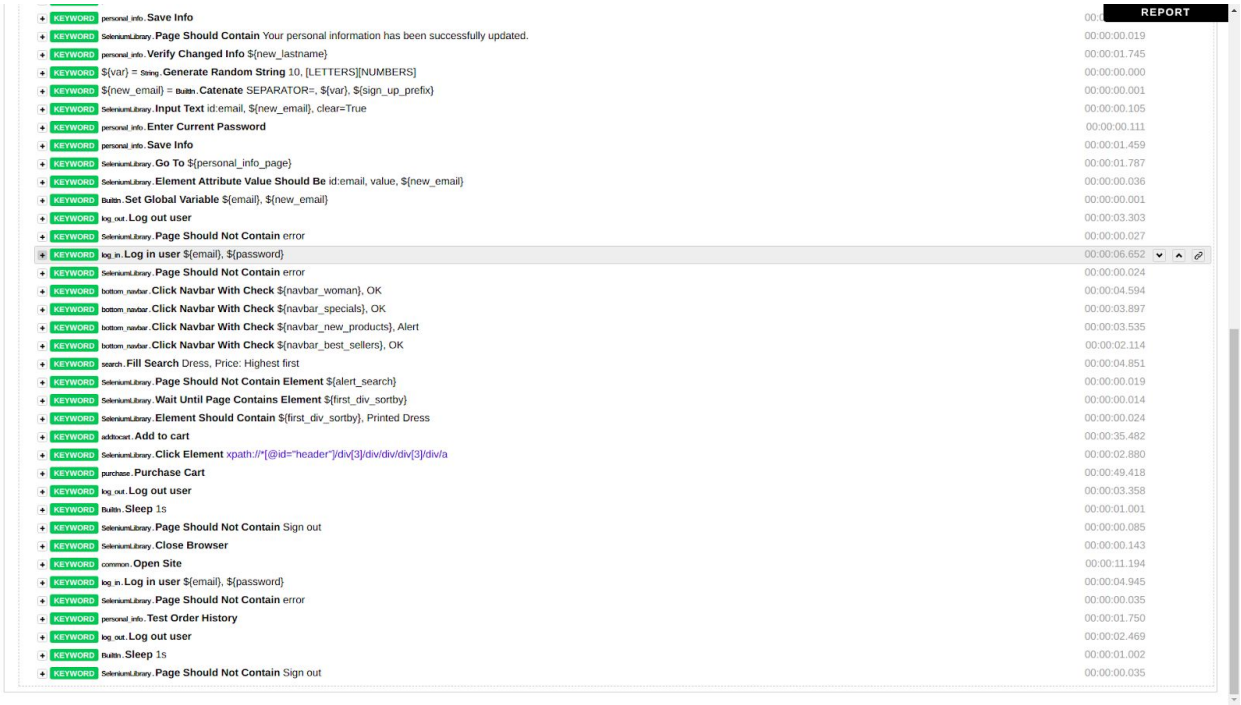
(Figure 1: The Running Process of the System Test from the Console.)

Output Files

The framework generates a set of output files that illustrate the test cases results:

1. output.xml: it contains all the test execution results in machine-readable XML format, from which the HTML files are generated.
2. report.html: a brief report that reports the passed test cases and failed test cases.
3. log.html: it includes all the details about each test case, the passed keywords, and if a failure exists. The keyword that causes failure and the error message.
4. screenshot.png: some libraries such as the selenium library provides a screenshot of the error if it occurs.





(Figure 2: Screen of the LOG file. That contains the keywords and functions run by the ROBOT Framework.)

Work Distribution

Team Member	Activities
Omar Ahmed omar.ahmed983@eng-st.cu.edu.eg	• Tested the Navigation component. • Tested the Search component. • Tested the contact_us component.
Kareem Osama kareemosamasobeih@gmail.com	• Tested the Home page. • Detected Flaws in the filtering system. • Tested the Compare component, and detected flaws in it.
Muhammad Sayed muhammad.mahmoud98@emg-st.cu.edu.eg	• Tested User Profile updates. • Tested Addresses component. • Tested Order History Component.
Evram Yousef evramyousef@gmail.com	• Tested Login and Logout • Tested Sign up process and detected flaws in it, as well as the integration with login and logout. • Tested the Purchasing Process. • Designed the interface of the project.

References

- On How to run the code, please refer to the readme.md file at our repository:
  https://github.com/Evraa/Testing-Project-Robot_Framwork
- The SUT:
  http://automationpractice.com/index.php
- ROBOT Framework:
  https://robotframework.org