Automated Testing Documentation of the Mobile App Control Centre

Google Summer of Code 2015
Daisy Nkweteyim

Project Abstract

The development of this project includes setting up a generic automated testing framework in which the Mobile App Control Center, a Peace Corps project which the Systers community is undertaking, can be tested. In addition, database tests will be implemented to automate queries to fetch data from the database. Selenium(Junit testing framework) implemented in Java will be used. The end product should be tests which can be reused and automated for regression testing.

Background Information

The Peace Corps has many projects which have been started to help the Peace Corps volunteers in their assignments. Systers will be working on some of these projects with the Google Summer of Code (GSoC) program so these project can be quickly deployed to the volunteers who need the technology. Some of the projects are:

PeaceTrack - Android and iOS Version

Malaria Prevention (Android and iOS versions)

Mobile App Control Center (Python w/Django)

The PeaceTrack and Malaria Prevention are both mobile app while the Mobile App Control Center (MACC) is the web application that will manage the data of both the PeaceTrack and the Malaria Prevention apps. The aim of automated testing of the MACC is to build an automated test framework for the MACC.

Tools Used

Ubuntu 14.04 LTS
Eclipse IDE for Java Developers (Version: Luna Service Release 2 (4.4.2))
Java Programming Language
PostgreSQL Driver
Framework: Selenium
Build Tool: Maven

Installation Guide

Detailed instructions for getting this project set up can be found here:
https://github.com/systers/automated-testing/blob/develop/MACCTests/README.md

In the case where the default username and password provided is not used, go into the project in Contants.java under utilities, enter in your preferred username and password in defaultUsername and defaultPassword respectively.

To run all the test cases, run Main.java found in src/utilities.

Code Structure

Here is a breakdown of the folder structure of the project:
The src folder contains the following packages: common, data, database_tests, home_page_features, post_features and utilities.
The **common package** contains two classes; CommonCode and DataProviderCommonCode. Both are TestNG classes which provide common functionalities used by the other classes. They are mainly used as super classes. Their main function is to load the homepage of the MACC and close the driver after the tests have run. The difference between the two classes is that DataProviderCommonCode logs into the MACC for the test cases that require login credentials while CommonCode does not log into the MACC for those test cases that do not requires login credentials. An example is the signup test case.

The **data package** contains an excel spreadsheet of the xml spreadsheet format (.xlsx). This spreadsheet contains five (5) sheets of test data with each sheet containing test data for different test cases.

This table shows the tests cases that the sheet number corresponds to.

Sheet Number	Test Case
Sheet 1	Login test case
Sheet 2	Signup test case
Sheet 3	Add post and edit post test cases
Sheet 4	Change Password test case
Sheet 5	Edit Profile test case

The **database_tests package** contains tests for the database of the MACC. It essentially tests the MACC in conjunction to its database to make sure the actual data matches the expected data when CRUD (Create, Read, Update, Delete) operations are performed on it.

The **home_page_features package** contains tests cases which pertain to the home page features of the MACC. These include but are not limited to login, logout, signup, change password test cases as well as a test case to check all the links on the homepage to make sure they are functioning properly.

The **post_features package** contains test cases which have to do with posting in the MACC. This includes adding and deleting a post.

The **utilities package** contains the excel reader class which reads the data from each sheet of the excel spreadsheet. The data read from these sheets get passed to the test methods through TestNG @DataProvider annotation. Each test case which uses the data from the sheets therefore gets as many times as there are test data.

This package also contains testng.xml and Main.java. Testng.xml provides all the test cases as a test suite to be run. When Main.java is executed, it call testng.xml which runs all the test cases. The output of these tests are in a file called TestNG Test Suite.

References

- http://web.archive.org/web/20110707113430/http://info.allianceglobalservices.com/Portals/30827/docs/test%20automation%20framework%20and%20guidelines.pdf
- http://www.softwaretestinghelp.com/selenium-tutorial-1/
- http://university.utest.com/test-case-writing-creation/#Approachtestcasewriting
- http://en.wikipedia.org/wiki/Test_automation
- http://www.ontestautomation.com/create-your-own-html-report-from-selenium-tests/