Demo app

Documentation

V1.0

Author: Rashi Nagar

Table of Contents

About Demo App..... 3

Testing Approach 3

Test Coverage 3

How to Run the Application Test Suite 4

Report of Executed Tests.....6

Report of found issues/bugs.....6

Answers about testing and testability.....7

Improvement for the app/ Potential gaps in Application.....9

If you would be given a week to do quality assurance for this product, briefly plan the tasks based on your skills, knowledge and expertise 11

How much time it took? 12

Demo web App

About Demo web app

Demo web app is a simple login application that allows user to Register and then login to the application where user can view its information.

Testing Approach

I have used Python Robot Framework to create Automation Test Suite for UI and Python unit Test Suite for API testing of the Demo app Application.

This application has 4 major functionalities:

1. Registration
2. Login
3. View user information
4. Logout

Test Coverage

1. **UI Test Suite**

   TC001_New_User_registration : This test case ensures that every time a new user is successfully registered.

   TC002_Login_and_Verify_user_info_after_successful_registration_for_new_user: This test case ensures that the user which we registered in "TC001_New_User_registration" is able to successfully login, all the details appearing in the user information section are of the logged in user and the user is able to successfully logged out of the application.

   TC003_Verify_Already_Registered_user : This test ensures that a user cannot re-register itself.

   TC004_Invalid_Login: This test case ensures that user is not able to login to the app if user enters invalid username or invalid password or both the details invalid.

2. **API Test Suite**

I have written individual functions with required arguments to call individual API's. I am using these API endpoints in the Unit test framework with static test data. Below are the details of Unit tests:

- **test_api_auth:** This test validates, for a registered user auth should return a valid token.
- **test_api_users:** This test validates, a registered user should be able to get the list of all users in the system.
- **test_api_username_get:** This test validates, a registered user should be able to see its information.
- **test_api_username_put:** This test validates, a registered user is able to update its information.

How to Run the Application Test Suite:

1. **Prerequisite**:
   - Python 3
   - SQLite3
   - Open command Prompt, run it as Administrator and Install following if you don't have it already:
     - Robot Framework
       command: **pip install robotframework**
     - Robot Framework-Selenium Library
       command: **pip install robotframework-seleniumlibrary**
   - Install Chrome Browser and its compatible Chrome Driver. Below is the location from where chrome driver can be downloaded.

     https://chromedriver.chromium.org/downloads

2. To test the Demo app using this test suite first run the demo app on local host at port number 8080.

3. Please register a user with below details:

Username	apiuser
Password	test@123
First Name	Test
Last Name	Api
Phone Number	45672389

4. Open Command Prompt and go to solution root folder and execute the following:

   Below is the command to execute UI Test Cases
   - robot TestCases\UI_Test_Cases.robot

   Below is the command to execute API Test Cases
   - APITestCases\test_cases.py

5. Open "Automation\Signant_Health\report.html" in a browser for the detailed information on the execution of test suite.

Report of Executed Tests

REPORT

UI Test Cases Log

Generated
20200821 14:27:20 UTC+03:00
1 minute 13 seconds ago

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	4	4	0	00:00:50	
All Tests	4	4	0	00:00:50	

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
Invalid data verification	2	2	0	00:00:25	
Valid data verification	2	2	0	00:00:25	

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
UI Test Cases	4	4	0	00:00:50	

Test Execution Log

SUITE UI Test Cases 00:00:50.357

Full Name: UI Test Cases
Source: C:\Users\rashi\PycharmProjects\SignantHealth\Automation\Signant_Health\TestCases\UI_Test_Cases.robot
Start / End / Elapsed: 20200821 14:26:30.474 / 20200821 14:27:20.831 / 00:00:50.357
Status: 4 critical test, 4 passed, 0 failed
4 test total, 4 passed, 0 failed

- TEST TC001_New_User_registration 00:00:13.671
- TEST TC002_Login_and_Verify_user_info_after_successful_registration_for_new_user 00:00:11.784
- TEST TC003_Verify_Already_Registered_user 00:00:12.234
- TEST TC004_Invalid_Login 00:00:12.297

Report of found issues/bugs

I am using Python 3 to run demo app and I faced an issue while using PUT api. In python3 it seems *iteritems()* has been deprecated and we can use *items()* instead.

Answers about testing and testability

1. **How do you Code review and enforcing coding standards?**

    - I look for the correct Indentation
    - Check for if Agreed Upon Naming convention has been used
    - If there is any piece of code that we can use, I suggest to reuse that code
    - I suggest to write Unit test if it is not there already
    - Code should be easily readable
    - I check if the names of (variables, parameters, methods and classes) actually reflect the thing they represent.
    - I check if I can understand what the tests do
    - I check if the tests cover a good subset of cases along with the corner cases. Are all the happy paths covered, I check if there are any cases that haven't been considered in the tests.
    - I check if the exception error messages are easy to understand and are meaningful.
    - I check if documentation is done for the test cases.
    - I check if the code actually does what it is supposed to do.

2. **How do you plan what kind of approach you take for test automation - what libraries to use, how does it work in couple of years, how to make it easy to maintain, etc? What are the main points to consider?**

    - Test cases should be simple and easy to understand.
    - Test cases should be comprehensive which includes all the corner test cases.
    - Test cases should be written at module level so that in case of any requirement change in future, respective module can be updated without impacting other functionality.
    - Usage of widely used libraries is done so that chances of unexpected behaviour is less.
    - I consider backward compatibility as well to make my code workable in future as well.

- Test suites should be written in a way so that new test cases can be easily added.

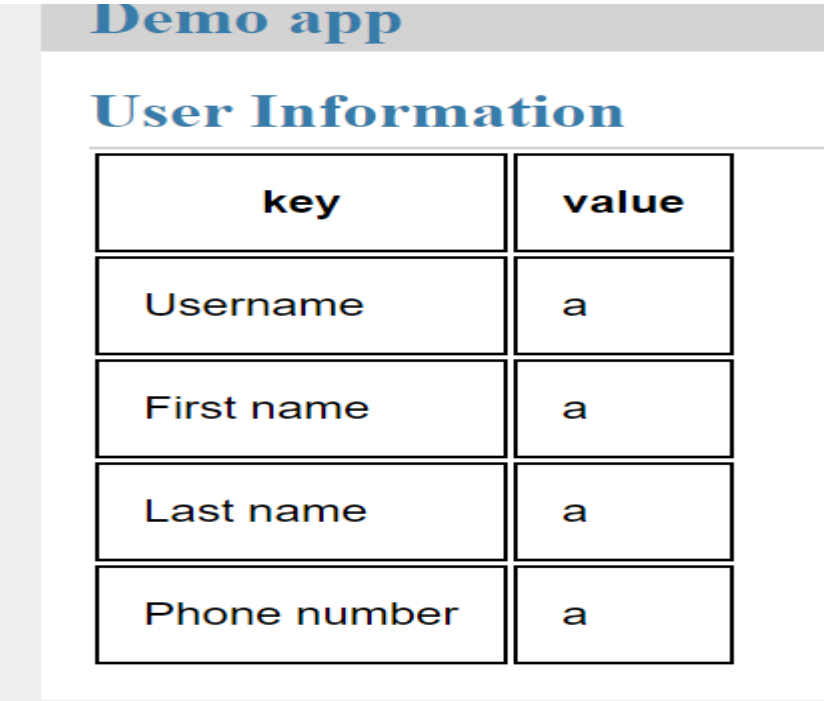
3. **Code testability, how do you enforce it? How do you make sure that the product is testable?**

   To make a product testable:

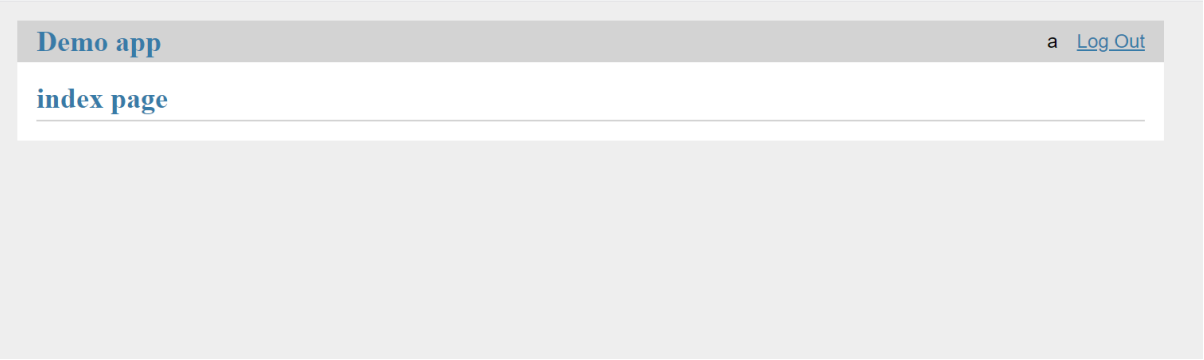
- Requirement should be clear
- Requirement should be feasible
- Requirement should be consistent
- Requirement should be complete
- Requirement should be unambiguous
- Requirement should be verifiable in real environment
- For every requirement we will create test cases and test data related dependency should be resolved.
- A checklist should be created and followed.

Improvement for the app/ Potential gaps in Application

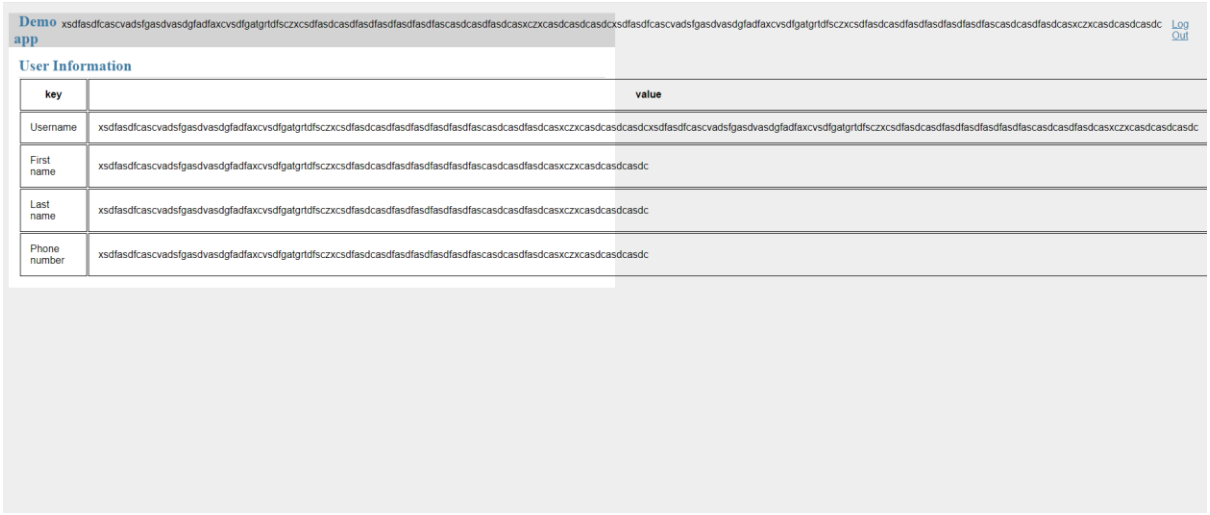
- Validation should be provided on all the fields. For eg: Phone Number field is accepting alphanumeric characters, First Name, Last Name, Username are accepting 1 alphanumeric character.



- On user information screen when user clicks on “Demo app” link it navigates to index page and user information can only be viewed after logging out and login again the application. User information should be visible once the user log in the application and on navigation there should be a way to view the user information again.



- UI gets distorted on entering large characters in input fields during registration. Restriction on characters should be provided to avoid screen distortion.



If you would be given a week to do quality assurance for this product, briefly plan the tasks based on your skills, knowledge and expertise

1. **Planning**: The scope of testing along with "Must have" requirements i.e. Browser, version, platforms should be clearly defined. Team should be skilled or trainings should be provided as per the requirement of the project. Risks should be identified along with their Mitigation strategy. Tools should be identified.

2. **Quality Audit**: A well-defined quality audit process to be implemented and followed to maintain the quality standard of the product under release.

3. **Reviewing the Epics/ User Stories**: Firstly, all the User stories should be reviewed thoroughly, questions should be noted and discussed within the team. If there are any gaps found in the understanding of the team then it should be further discussed with Business Analyst or authorized person and gaps should be eliminated. Also, test cases should be updated on timely basis to avoid pesticide paradox.

4. **Feasibility study of Automation Test cases**: Test cases should be identified and feasibility study should be done.

5. **Test Case creation**: All the test cases/ Test scripts should be created considering all the positive and negative scenarios.

6. **Smoke Test Cases**: Smoke testing checklist should be created and should be executed once the application is release on QA server.

7. **Acceptance Test Cases**: Acceptance test cases should be created which will be executed once the build is released with the new functionality under test and has passed all the Smoke test cases. In case of any minor test case failure which is not a blocker for testing we can proceed and issue can be raised on defect management system.

8. **Traceability Matrix**: Traceability Matrix should be created to see the coverage of all the test cases with respect to the user story.

9. **Test Environment Setup**: The application should be available on QA server; DB connectivity should be proper and ready for testing purpose.

10. **Test Case Execution**: All the Smoke and Acceptance test cases followed by the regression suite should be executed.

11. **Exploratory Testing**: This type of testing helps the testers to catch those errors which were missed while test execution. Also, this testing can only be done once the tester has good understanding of the application.

12. **Defect reporting**: Issues found during test case execution should be reported in defect management system following the complete format of logging defects and supporting it with screenshots and videos. A defect should include Timestamp, Platform/ Browser/ Version.

13. **Retesting**: When fix is provided by the development team, retesting of the defect should be done.

14. **Decision on Critical defects**: Root cause analysis of critical defects should be done; fix should be provided as per of intermittent build to avoid any delay in production release.

How much time it took?

I am submitting the solution after exactly in a week but I will measure my efforts that I have put in to complete this assignment as 3 Story points.