**QA Task**

1. **Test Cases**

Tests Cases for registration and login forms. Test cases from this document are covered by python’s function. Each python function covers at least one of test case.

**1.1 Sign up cases:**

**First step before registration form is visible**

TC1: Check if Email fields is visible, automated in test\_case1, prio1

TC2: Check if frames of fields are highlighted in red if it is empty, automated in tect\_case1, prio3

TC3: Check if combobox is expended down when cursor is inside, automated in test\_case8, prio1

TC4: Check if frames of fields are highlighted in blue if they are filled as expected, automate in test\_case1, prio3

TC5: Click create an account without email, automated in test\_case4, prio2

TC6: Click create an account with invalid email address, automated in testcase5, prio2

TC7: Click create an account with already used email address, automated in test case 3, prio1

TC8: Click create an account with good email address, automated in test case2, prio1

**Second step when full registration form is visible**

TC9: Check if all fields are visible in form: first name, last name, password, automated in test\_case6, prio1

TC10: Check if all required fields has star stamp, automated in test\_case6, prio1

TC11: Check if tab work as expected, to do, prio3

TC12: Check if enter work like left mouse button, to do, prio3

TC13: Check if all placeholder is empty by default expect this that were filled previously, automated in test\_case6, prio2

TC14: Check if it is confirm password field, automated in test\_case6, there is no confirm password field in form, prio1

TC15: Check if password is encrypted, automated test\_case8, prio1

TC16: Check if frames are highlighted in blue when they are filled as expected, automated in test\_case7, prio3

TC17: Check if frames are highlighted in red when they are empty and cursor is in field, automate automated in test\_case7, prio3

TC18: Check if data processed to the server after filled all fields, automated in test\_case8, prio1

TC19: Check if it was successful registration after all fields were filled except this without required star stamp, automated in test\_case9, prio1

TC20: Check if it was unsuccessful registration after left all fields empty, automated in test\_case7, prio1

TC21: Check if it was unsuccessful registration after left empty field with required star stamp, automated in test\_case10, prio1

TC22: Check if registration is unsuccessful with password less than 5 position, automated in test\_case11, prio1

TC23: Verify that digits in last name and first name cause an error automated in test\_case11, prio2

TC24: Check if "Sign up for our newsletter!", "Receive special offers from our partners!

" works are clickable, automated in test\_case7, prio2

TC25: Check if registration is unsuccessful with wrong format of mobile phone, automated in test\_case11, prio2

TC26: Check if registration is unsuccessful with wrong format of postcode, automated in test\_case11, prio2

TC27: Check if there is not difference with letter size in all fields, automated in test\_case12, prio2

TC28: Check if registration message is visible after click on register button, automated in test\_case7, prio2

TC29: Check if there is proper validation message, after left empty all fields, automated in test\_case7, prio1

TC30: Check if the system distinguishes capital and small letter in already used email, automated in test\_case13, prio1

* 1. **Sign in cases:**

TC1: Check if cursor is focus on email placeholder, automated in test\_case14, cursor is not focus in placeholder, prio1

TC2: Check if tab/enter work as substitute to mouse left button, to do, prio3

TC3: Check if form contains element like username, password, sign in button, forgot your password, automated in test\_case14, prio1

TC4: Check if frames of fields are highlighted in blue when cursor is inside, automated in test\_case1, prio3

TC5: Check if placeholder are red when it is empty automated in test\_case1, prio3

TC6: Check if history in fields are expanded down when cursor is in placeholder, tested in many testcase, it does not work in selenium script, prio2

TC7: Check if all fields are empty and unmarked by default, automated in test\_case14, prio2

TC8: Check if user is able to login with valid credential, automated in test\_case15, prio1

TC9: Check if user is unable to login with invalid credential, automated in test\_case16, prio1

TC10: Check if user is unable to login with proper username and invalid password, automated in test\_case17, prio1

TC11: Check if user is unable to login with invalid username and proper password, automated in test\_case18, prio1

TC12: Check if user is unable to login with proper username and empty password, automated in test\_case19, prio1

TC13: Check if user is unable to login with empty email and proper password, automated in test\_case20, prio1

TC14: Check if user is not able to login with terminated credential, cannot change password so cannot test this feature, prio1

TC15: Check if password is encrypted, automated in test\_case15, prio1

TC16: Check if password can be copied, to do (password can’t be copied), prio1

TC17: Check that encrypted characters in “Password” field should not allow deciphering if copied, can be tested manually (can’t be copied in this case), prio1

TC18: Check if user can log in with new password after changing password, can’t be tested, email was not sent, prio1

TC19: Check if user can't log in with old password, can’t be tested, confirmation email was not sent, prio1

TC20: Check if user is still log in after clicking refresh button, automated in test\_case24, prio2

TC21: Check if user is still log in after clicking back and forward button browser, automated in test\_case24, prio2

TC22: Check if user is still log in after clicking home button, automated in test\_case24, prio1

TC23: Check if user is really logged out after clicking log out button, automated in test\_case25, prio1

TC24: Check if user is logged out after logging out and clicking back button, automated in test\_case25, prio1

TC25: Check amount of attempt unsuccessful login, tested manually, it seems to be many times, to do, prio1

TC26: Check if space is not allowed in password automated in test\_case21, space is allowed in password in this form it seems to be an error, prio1

TC27: Check if user is still logged in after closing of explorer, automated in test\_case26, prio1

TC28: Check if after changing password, email is delivered and password work, email is not delivered, prio1

TC29: Check timeout of being login, to do, prio1

TC30: Check if log out redirect to log in page, automated in test\_case2, prio1

TC31: Check if login form is not vulnerable to SQL injection, to do, prio1

TC32: Verify whether Cross-site scripting (XSS) vulnerability work on a login page. to do, prio1

TC33: Check if validation field is as excepted when email field for retrieving password is empty automated in test\_case22, prio1

TC34: Check if all fields and button are clickable in retrieving password form part, automated in test\_case22, prio1

TC35: Check if confirmation email of changing password was sent, automated in test\_case23, was not sent, prio1

TC36: Check if password is case sensitive, automated in test\_case27, prio1

1. **Test Management:**

**2.1 Test cases divided into priorities:**

**2.1.1 Sign up cases:**

**Priority 1:**

TC1: Check if Email fields is visible

TC3: Check if combobox is expended down when cursor is inside

TC7: Click create an account with already used email address

TC8: Click create an account with good email address

**Second step when whole registration form is visible**

TC9: Check if all fields are visibled in form: first name, last name, password

TC10: Check if all required fields have star stamp

TC14: Check if it is “confirm” password field

TC15: Check if password is encrypted

TC18: Check if data processed to the server after filled all fields

TC19: Check if it was successful registration after all fields were filled except this without required

TC20: Check if it was unsuccessful registration after left all fields empty

TC21: Check if it was unsuccessful registration after left empty field with required star stamp,

TC22: Check if registration is unsuccessful with password less than 5 position

TC29: Check if there is proper validation message, after left empty all fields

TC30: Check if the system distinguishes capital and small letter in already used email

**Priority 2:**

TC5: Click create an account without email

TC6: Click create an account with invalid email address

TC13: Check if all placeholder is empty by default expect this that were filled previously

TC23: Verify that digit in last name and first name cause an error

TC24: Check if "Sign up for our newsletter!", "Receive special offers from our partners!

" works are clickable

TC25: Check if registration is unsuccessful with wrong format of mobile phone

TC26: Check if registration is unsuccessful with wrong format of postcode

TC27: Check if there is not difference with letter size in all fields

TC28: Check if registration message is visible after click on register button

**Priority 3**

TC2: Check if frames of fields are highlighted in red if it is empty

TC4: Check if frames of fields are highlighted in blue if they are filled as expected

TC11: Check if tab work as expected

TC12: Check if enter work like left mouse button

TC16: Check if the frames are highlighted in blue when they are filled as expected

TC17: Check if the frames are highlighted in red when they are empty, and cursor is in field

**2.1.2 Sign in cases:**

**Priority 1:**

TC1: Check if cursor is focus on email placeholder

TC3: Check if form contains element like username password, sign in button, forgot your password,

TC8: Check if user can login with valid credential

TC9: Check if user is unable to login with invalid credential

TC10: Check if user is unable to login with proper username and invalid password

TC11: Check if user is unable to login with invalid username and proper password

TC12: Check if user is unable to login with proper username and empty password

TC13: Check if user is unable to login with empty email and proper password

TC14: Check if user is not able to login with terminated credential

TC15: Check if password is encrypted

TC16: Check if password can be copied

TC17: Check that encrypted characters in “Password” field should not allow deciphering if copied, can be tested manually

TC18: Check if user can log in with new password after changing password

TC19: Check if user can't log in with old password

TC22: Check if user is still log in after clicking home button

TC23: Check if user is really logged out after clicking log out button

TC24: Check if user is logged out after logging out and clicking back button

TC25: Check amount of attempt unsuccessful login

TC26: Check if space is not allowed in password

TC27: Check if user is still logged in after closing of explorer

TC28: Check if after changing password, email is delivered and password work

TC29: Check timeout of being login

TC30: Check if log out redirect to log in page, automated in test\_case2

TC31: Check if login form is not vulnerable to SQL injection

TC32: Verify whether Cross-site scripting (XSS) vulnerability work on a login page. XSS vulnerability may be used by hackers to bypass access controls

TC33: Check if validation field is as excepted when email field for retrieving password is empty

TC34: Check if all fields and button are clickable in retrieving password form part

TC35: Check if confirmation email of changing password was sent

TC36: Check if password is case sensitive

**Priority 2:**

TC7: Check if all fields are empty and unmarked by default

TC20: Check if user is still log in after clicking refresh button

TC21: Check if user is still log in after clicking back and forward button browser button

**Priority 3:**

TC2: Check if tab/enter work as substitute to mouse left button

TC4: Check if frames of fields are highlighted in blue when cursor inside

TC5: Check if placeholder is red when it is empty

TC6: Check if history in fields are expanded down when cursor is in placeholder

**2.2 Test Cases Management approach.**

**2.2.1 Overall information.**

For the purposes of demonstrating how to manage tests, we assume that the development sprint is 2 weeks period, and an official build is released every two weeks to the client, developers provide the code daily, first to the development branch and after initial testing to the trunk, where tests are also run.

There are 2 repositories in the project, the first is where the programmer's code is added and the second is where the tester regression tests are committed, both repositories have a trunk and development branch as well.

When programmer starting the implementation of a new feature, tester should take care of creating the test in parallel, when the developer finishes creating the feature, test should also be completed, both the test and the developer feature may be committed to developer branches. If test is not ready on time, tester should test the feature manually before delivering it to trunk.

**2.2.2 When and where we run tests.**

**Priority 1:**

Regression testing with priority 1 should be ran with each delivery from developers, on branch and on trunk, in order to check the software after each delivery at the most critical points, if feature passes automatic tests with priority 1 and possibly manual tests, which have not yet were added to the regression, then developer can merge code to the trunk, while committing and merging it should be ran job on Jenkins with tests.

**Priority 2:**

Regression tests with priority 1 and 2 (the pool of tests increased by these with priority 2) should be ran, 2x a day on the trunk, in order to eliminate possible errors in less critical areas that could be caused by commitments from the whole day.

**Priority 3:**

Regression tests with priority 1, priority 2 and priority 3 (pool of tests with all priority) should be ran once a day in the evening and checked in the morning to verify the software in the all areas together.

**2.2.3 How to manage regression on Jenkins.**

Tests that are ran on Jenkins on both branches should be watched by at least one tester from the team, let's assume we have 2 testers in team, so they should exchange every week. If tester notices build on which tests are failed, he should attempt to investigation.

Tester should eliminate all external causes that could affect the result, i.e. work on servers, he should consult with the build coordinator. In the case that the tester eliminated all external causes, he should decide whether the error is in the software that is testing, or whether there are necessary changes in the test that were not agreed with the developers previously.

If the bug is in the code being tested, then a task should be issued to improve the code in Jira. If it is opposite, then task for test alignment to software should be created. In such task should be included all necessary information like, name of test, build number, test report etc.

In the case that rate of failed test is not 100%, it is e.g. 4/5 fails, then should be considered stability issue. More about stability is written in section “A bit about stability”.

**2.2.4 A little bit about adding test to regression.**

Before creating a review for code, test should be checked locally on the code it is testing and should also be ran 100x to check its stability. Test should undergo a review before adding to the regression, by 3 people, one person can be a tester from team, 2 person can be a technical leader from team and 3 person can be a tester from another team that deals with testing others part of the software, e.g. backend or database.

After taking the review, test should be added to the pool of tests that has the "candidate" tag, and with this tag can be committed to branch and then to trunk. The candidate tests pool are tests that are not taken into account in software regression tests, but are tested on this software, when is programmer’s delivery Jenkins job with candidate tests is triggered, and if in the next 5 builds on Jenkins, test has a passed result, then tag candidate may be removed and tests can be committed to the normal pool of validation tests. The same process applies to old tests that are being repaired.

**2.2.5 A bit about stability of tests.**

While tests run on Jenkins, if tests pass successfully, then they are ran only once, while tests fail, then they are ran 5 times if we have 5/5 fails, it means that there was a bug in the software, or something has changed in implementation and we need to apply patch for the test, if the test fails 3/5 times, it means that the test is unstable and should have an unstable tag added to it, which will eliminate it from the tests taken into account in regression pool, task in Jira on fixing this test should be reported then the test should be repaired, ran 100 time locally and if its 95% passed then can be added back to pool of regression, before going through the process of adding regression tests that I described above

**2.2.6 Other areas of tests.**

During delivery to trunk and other branch, other types of tests should also be ran, such as database tests or Restapi tests, so that we know if all application areas are valid.

**2.2.7 Acceptance test.**

Before releasing the build to the client, acceptance tests should be passed, i.e. maximum number of tests from each area of an application, after passing a certain percentage it is usually 95%, build tests can be marked as release and provided to customer.

1. **Test Strategy**

**3.1 Responsibilities and Objectives.**

* **Author:** Jakub Stawowczyk
* **Reviewers:** Sofomo Testing Team
* **Approvers:** Sofomo Management Team

The purpose of this document is to prepare a test strategy for testing of form with registration and login form on the website <http://automationpractice.com/>

* 1. **Testing approach:**

**3.2.1 Process of testing:**

* Test planning
* Test analysis
* Writing test scenario
* Performing manual tests
* Performing automated tests
* Maintenance of tests and tests documentation

**3.2.2 Testing areas:**

**Form layout:**

Layout of each form should be easy to understand for users, so it is very important to perform usability testing to avoid discouragement of users. Form with many fields is difficult to fulfill and uncomfortable for users to.

**Form fields:**

Form consists a lot of different type of field and, tester should check all of them if it is possible to use them, for example write text in textbox or select proper option in all fields

**Required Fields:**

Fields that are mandatory should have asterix next to them and should be checked out if they are mandatory in real. It means that all required field should be filled to successfully process data to server. All field with asterix should be tested, it means that each mandatory field should be omitted singly in testing negative scenario.

**Input validation:**

Form should validate input on some fields, such as email address, zip code fields, phone fields. If an alphabetic are entered into phone number field then well worded validation message is helpful for user to understand error.

**Validation message:**

Verify that validation message correctly describes the problem, such as whether the field is required or if input is incorrect.

**Other validation elements:**

Some fields are blue or green if are filled correctly or red if are filled incorrectly. It should be checked in test.

**Tab and Enter check:**

Verification of tab and enter button that they are work as should be.

**Prefilled fields:**

Some of fields may be filled by default

**Thanks Page**

Once the form has submitted, check that a thanks page is displayed to the user with a relevant thank you message.

**Email Confirmation:**

Verify that email confirmation was sent if it should be sent,

Verify that confirmation link will activate account

**Form action:**

Check that data was successfully saved in database.

**Security tests:**

SQL injection

XSS testing

**Database tests:**

Check that data can be added and removed from database using form

**RESTAPI tests:**

Check that endpoint work as expected in e.g. Postman

**Integration tests:**

Tests between Database, RESTAPI, and application form

**Browser tests:**

Tests should be executed on different browser e.g. google, internet explorer

**OS tests:**

Test should be executed on different operating system.

**Mobile tests:**

Test execution on mobile phone.

**3.2.3 Types of tests that should be executed:**

**Manual tests:**

Before automation test should be done manually

**Black box tests:**

All cases from chapter 1 are black box tests

**Functional tests:**

We test “what” application do

**Load tests:**

We can check responsiveness and reliability of form used by many users in the same time

**Usability tests:**

Form layout tests

**Compatibility tests:**

Check test on different explorer

Check test on different operating system

**Security tests:**

SQL injections test

XSS test

**Stability tests:**

Run test in many iterations to check if its stable

**Automated test:**

Test in Selenium using Pytest framework

* 1. **Exit Criteria**
* Time required to finish all tests and documentation: 24h
* Test that should be run: 100%
* Test that should be passed: 95%
* Functional coverage: 100%
* Identifying and fixing all the high-priority defects
* Identifying and fixing all blockers and high-severity defects
* All tests logs, test incident report logs, tests summary report should be collected

**3.4 Test environment and tools.**

* Github
* Python
* Selenium
* Pytest
* Pytest html raport

1. **Reference**.

My