**Suppose, as an SQA engineer, you need to test the ‘Username’ field of a ‘Registration’ form, which allows 6 to 12 alpha-numeric characters only.**

***(a) What will be your test plan/strategy to test this field?***

The test plan/strategy to test the username field is given below:

1. Scope of Testing:

1.1 The features need to be tested:

* UI interface: The field needs to be user-friendly, findable, and self-explainable. The username should be marked \* over the input field (as most of the time it is mandatory information). Furthermore, there should contain a valid placeholder in the username field.
* Error message handling: While giving any invalid input, typing a username that already exists, leaving the field blank, or giving less than 6 characters and more than 12 characters should show a validation error.
* Input box: Copying and pasting any character and tab functionality need to operate accurately.
* Browser interface: The registration form can display without any break in all different kinds of browsers.

1.2 The features do not need to be tested:

* Operating system interface: As the registration form will run in the browser, it does not depend on any OS.
* Communication interface: As it is not mentioned in the requirements.

2. Test type: Unit testing, integration testing, system testing, reliability testing, usability testing, compatibility testing.

3. Test objective: The objectives are to check the functionalities of the username field in the registration form. For example, uniqueness in the username, valid characters identification, the limit of characters, showcase of a validation error, browser compatibility, etc. should work perfectly every time.

4. Suspension Criteria: If it is discovered that 50% of the test cases are failed, reject the rest of the testing until the developers fix all the failed cases.

5. Resource tool: Selenium, Katalon Studio.

6. Duration to complete the task: 2 hours.

7. Test deliverables: Test case documents, test reports, defect reports.

***(b) What will be your test cases to test this field?***

Ans:

1. Verify that the username is marked \* over the field (as most of the time it a mandatory information).

2. Verify that the space is not allowed in the username field or send a validation error message if space is given.

3. Verify that system generates validation error while entering characters which is not alpha-numeric characters (A to Z, and 0 to 9) in username field.

~~3. Verify that system generates validation error if any symbol is used in username field.~~

4. Verify that system generates validation error while entering more than 12 characters in username field.

5. Verify that system shows validation error while entering less than 06 characters in username field.

~~5. Verify that system shows no validation error if the number of characters is from 6 to 12.~~

6. Verify that system displays validation error while the username field left blanked.

~~6. Verify that system displays no validation error while the username field is filled.~~

7. Verify that system gives validation error while entering an existing username.

~~7. Verify that system gives no validation error while entering a unique username.~~

8. Verify that system generates no validation error for case sensitivity.

9. Verify tab functionality is operating accurately.

10. Verify the username field contains valid place holder.

11. Verify that the ‘save username’ checkbox is unselected by default.

12. Verify that the registration form is displaying without any break in all different kinds of browser.

13. Verify that the link of the registration form is operating fine even if lots of users fill the username at the same time.

***(c) What types of testing will you execute for that field?***

I would execute both functional testing and non-functional testing for that field.

Functional testing:

1. Unit testing: Validate that the username field is taking input and showing the expected outcome. Giving more than 12 characters or less than 06 characters should not be accepted and would give an error message. Check by providing any kind of symbol in the username field generates an error.

2. Integration testing: Expose defects in the interaction between modules in the registration form. For instance, if a user gives a valid username (as well as other required information if any), and clicks submit button then it will be redirected to a successful submission page. As a tester, I would check the interface link, data communication between these two pages.

3. System testing: Check after integrating all the fields, links, images, checkmarks in the registration form, the username field is working as expected. Both black box and white box testing techniques can be followed for this.

Non-functional testing:

1. Reliability testing: Perform a function for a long period such as filling up the field and clear it continuously for 1 hour in a bad internet connection, and check either the software application crashes or fail. I would use automation tools for this testing.

2. Usability testing: The name of the field, error handling message, input box should be user-friendly, findable, and self-explainable. Such as, while writing a username, the validation message continuously shows that either the number of characters’ limit is crossing or not, suggests an existing username, generates a message if the user name is already taken, etc.

3. Compatibility testing: Check that the registration form can be filled up by opening from any browser, device, mobile, and network. I would test it by opening from browsers like Firefox, Google Chrome, Safari, Microsoft Edge, etc. Testing can be done by filling up using both PC and mobile devices.

For testing the username with input data, I can follow four software testing techniques:

1. Boundary value analysis: For testing the input values in the username field, this technique can be followed. It is based on taking a maximum value, minimum value as well as some inside and outside values of a boundary. As the limit of valid input characters is 6 to 12 so, the boundary is 6-12 and I can take 5, 6, 10, 12, 13 numbers of characters as input of the username.

2. Equivalence class partitioning: It will divide the input values into equivalent classes. One value from that class can be used for testing and it will consider as same for system behavior. So, the classes of input value for testing the username field could be 0-5, 6-12, 13-20.

3. Positive testing: Providing valid input (only 6-12 alpha-numeric characters) in the username field and check whether the registration form is accepting it.

4. Negative testing: Test the registration form by giving invalid input in the username field and check the system gives an error message. For instance, using mathematical symbol (?, >, <, +, =, /, %), giving more than 12 characters and less than 6 characters in the username field.

***(d) What are the input values that you will use for testing this field?***

I would select the test inputs based on the four test techniques that I mentioned above.

1. Boundary value analysis: moin2 (shows error), moin25 (accept), moinul2523 (accept), 27moinul2625 (accept), moinul2526272 (shows error).

2. Equivalence class partitioning: mo25 (shows error), Moinul25 (accept), 30MoinuLIslaM31 (shows error).

3. Positive testing: moinul27, 25moinul, moi27nul, 25262721272m, MOINUL27, MoInUl27 etc.

4. Negative testing: moin, MOIN, moinulislamshawon, MOInulIslamSHawoN, MOINUL2777777, moin+, moinul+, moinul+27, MoinuL+27, 2+7moinul, (MOINUL), MOINUL?, moin\*moin, moin&moin, moin>moin, moin/moin, ^&\*()<>?/, etc.