**TEST CASES DESCRIPTION**

**New User Form Positive Tests** (*NewUserFormPositiveTests.java class*)

1. **checks that users with diferent options are created and user info on users page is correct:**

**case 1:** only mandatory fields are filled in (login/password/repeat password)

**case 2:** mandatory fields + 1 optional (full name) are filled in

**case 3:** mandatory fields + 1 optional (email) are filled in

**case 4:** mandatory fields + 1 optional (jabber) are filled in

**case 5:** mandatory fields + all optional (full name/email/jabber) are filled in

1. **checks creation of user with minimal fields length** -> *GREY AREA BY REQUIREMENTS, MIN LENGTH IS NOT SPECIFIED AND ACCEPTED AS 1 SYMBOL*

**case 1:** create user with minimum fields length (all fields = 1 symbol, except login -> *exception made to be able to find user later and check created user info, as 1 symbol doesn’t provide unique search result)*

**case 2:** create user with minimum fields length (all fields = 1 symbol, except full name -> *exception made to be able to find user later and check created user info, as 1 symbol doesn’t provide unique search result)*

**case 3:** create user with minimum fields length (all fields = 1 symbol, except email -> *exception made to be able to find user later and check created user info, as 1 symbol doesn’t provide unique search result)*

1. **checks creation of user with maximal fields length** -> *GREY AREA BY REQUIREMENTS, MAX LENGTH IS SPECIFIED ONLY FOR LOGIN ANF FULL NAME AS 50 SYMBOLS, FOR ALL OTHER FIELDS NO LIMITATION*

**case 1:** all fields length = 50

**case 2:** all fields length > 50 (but login and full name are auto-truncated till 50 symbols)

1. **checks creation of user with special symbols in all fields** GREY AREA - REQUIREMENTS ARE NOT SPECIFIED - TEST FAILS ON USER INFO ASSERTION

**case 1:** for special symbols *"!±@#$%^&\*()-\_=+{}[];:\"'|\\<>,.?/~`"*

shows that they are allowed in every field (only login doesn't allow "<>/" and space)

*\*corresponding negative test for login with special symbols is in negative tests*

1. **checks that user is not created when clicking Cancel on New User Form**

**case 1:** fill in new user form data and click Cancel button. Ensure that user wasn’t created

**New User Form Negative Tests** (*NewUserFormNegativeTests.java class*)

1. **Checks that user with empty mandatory fields is not created:**

**case 1:** login field is empty

**case 2:** password field is empty

**case 3:** repeat password field is empty

1. **Checks that user with inconsistent info is not created:**

**case 1:** values in password and repeat password fields don’t match

1. **Checks that user with specific special symbols is not created: ->** CREY AREA, as CAN SEE FROM CORRESPONDING POSITIVE TEST MOST OF SPECIAL SYMBOLS ARE ALLOWED IN ALL FIELDS

**case 1:** login name contains space

**case 2:** login name contains “>”

**case 3:** login name contains “<”

**case 4:** login name contains “/”

**case 5:** email contains space -> FAILS, NO EMAIL VALIDATION

**case 6:** jabber contains space -> FAILS, NO JABBER VALIDATION

1. **Checks that user with wrong email/jabber format is not created\*:**

**case 1:** email with wrong format *!= Username@domain.extension*, e.g. "*345453767testemail*" -> FAILS, NO EMAIL VALIDATION

**case 2:** jabber with wrong format *!= Username@domain.extension*, e.g. "*345453767testjabberl*" -> FAILS, NO JABBER VALIDATION

1. **Checks that user with duplicated data cannot be created:**

**case 1:** create user with the same login name

**case 2:** create user with the same email -> FAILS, *however email shouldn't be duplicated as it's used for login functionality*

**\****cases for email/jabber format could be extended to check allowed special symbols according to email/jabber name agreements, but validation doesn’t work anyway even for spaces in email/jabber, so I skipped this part*

**Create User and Try To Login Positive Tests** *(NewUserLoginPositiveTests.java class)*

**case 1:** create User and try to log in with his login name

**case 2:** create User and try to log in with his email

**case 3:** create User with checked “*Force password change on first login*” checkbox and check that password is really forced to be changed on 1st login for this user

**case 4:** create User, login with his credentials and check that User Profile Page page shows correct user info

**Create User and Try To Login Negative Tests** *(NewUserLoginNegativeTests.java class)*

**case 1:** create User and try to log in with his login name and wrong password

**case 2:** create User and try to log in with his email and wrong password

**Permission To Create User Test** (UsersPageAccessTests.java class)

**case 1:** create user without admin permissions, log in with his credentials and try to access Create Users Page

**Create User and Search for It Tests** (SearchOfCreatedUserTests.java class)

1. **Search cases by full string:**

**case 1:** created user can be found by login name and his info matches with created user

**case 2:** created user can be found by email and his info matches with created user

**case 3:** created user can be found by full name and his info matches with created user

1. **Search cases by part of login, email, full name:**

**case 1:** search by partial login name (part of string from the beginning)-> FAILS

**case 2:** search by partial login name (part of string from the middle) -> FAILS

**case 3:** search by partial email (from begin till @)

**case 4:** search by partial email (from the middle) -> FAILS

**case 5:** search by partial full name (for parts separated by space)

**case 6:** search by partial full name for part not splitted by space ->FAILS