Testing report

|  |  |
| --- | --- |
| Types of testing techniques | Used? |
| Unit tests | Yes |
| Integration tests | Yes |
| User tests | Yes |

# Unit tests

In our project We mainly have 2-unit testing classes. In the first class we test methods related with the user, since nearly all the methods are wrapped around user related queries. The user related queries include creation of the table, user adding, user deleting, user password changing. Besides these queries we also have access levels, adding or deleting organization accesses, adding, or deleting event accesses. All these queries are the main methods used in our web app. So, we have prepared scenarios for these queries and tested them in Junit. For the tests we have created a new schema in the postgre Database. We called the schema junit. The one problem we encountered while preparing these tests were that because in junit when we do queries, since it does not actually create or alter anything, the tests were working but giving error that these databases do not exist. The reason was we created the databases in @Before method that runs once then all the @Test ‘s follows. And these annotations mean they only test, they do not actually do create tables. So that is why, in our junit class we skipped the exception that was giving “no table named … in database “error.

Another Junit class that we did was about Encryption. We thought that proving that our encrypting methods work properly would stop client’s password authentication worries. In the Encryption tests the first thing that we tested was that we can get validation for same password with same salt. We stored 2 strings with same hashing process, then we proved that they are the same. Another thing that we tested is, we proved that without the correct salt you cannot get the correct hash value. And finally, we proved that hashed password with salt and hashed password with salt that we went through encryption and decryption algorithm are the same passwords. Meaning encryption and decryption algorithms work.

# Integration tests

For the Integration tests we tested main model classes and their connection to MailDao and UsersDao classes. Since these classes were the only main classes that used other class objects, we created tests that checked getModel() method of these Dao classes. In the first test we tested the message class object and how it is filled with the help of query similar to the one used in the mailDao class. After filling we tested if it filled the message object correctly. We did the same test to usersDao class test, we created a user object and tested how it is filled with the help of a query method.

# User tests

User Test 1

**Testing procedure**

We asked our roommates to perform the following set of 10 tasks to test our applications usability:

1. Log In with an Admin account
2. Create a new user
3. View a mail
4. View a document
5. Download a mail/document
6. Create a new user
7. Add a group/event to the user
8. Add a group/event to a mail
9. Delete the user
10. Log Out
11. Change the password of an account

We chose roommates as our test group, because they fit the target audience of the app. They are all students, both Dutch and from abroad, who fit the general profile (in age, studies, etc.) of the users of our app.

**Feedback**

The feedback we got was:

1. The App needs to give more feedback, for example, when you delete a user, there should be a message that a user was deleted.
2. Refreshing the search after every keypress or making a search button would greatly help the UX in that part of the app.
3. Spamming the search button makes the app very slow.
4. The alerts are annoying.
5. There should be an option to make multiple users, or you should be redirected after making a user.
6. Some document links don’t work.
7. There is no profile, as a user you don’t know who you are.
8. Generalizing add and delete buttons for events and group.
9. Providing a faulty username causes issues.

**Conclusions**

In conclusion, we fixed the bugs that was caused by faulty usernames, we also fixed some other bugs that was caused by the system. We gave notification when deleting user. We added users name on pages so that it would resemble a profile.

User Test 2

Test product: kick-in project

Test target : to find bugs

Test task :

1. Log in
2. Make a new user
3. View a document
4. View a mail
5. Download a document/mail
6. Create a new user
7. Create a new group/event
8. Add an event/group to a user
9. Add an event/group to a mail
10. Log out
11. Change your password

Test summary:

1. It is difficult for users to find the password modification page
2. Can still return to the search interface after logging out
3. The attachment cannot be downloaded at the same time using the download button.
4. Some messages are not classified
5. The user wants to download multiple files at once
6. It will be more convenient if you set the access to the user when creating it.
7. Users want search results to be classified and displayed
8. The user wants to be able to modify the e-mail address.

Conclusion

We only were managed to solve some of the problems. However the changes the user has requested were not in our requirements so we decided to keep our version.

User Test 3

**Log In as an admin**

It was very familiar to any other login page making it easy to use.

**Register a new user**

Also, familiar design for easy use, though a redirect to the home page would be appreciated. There is also no menu button or button to go back.

**View a document**

The grid on the page made it easy to view specific documents.

**Search for a document that mentions “OKIC”**

Easy to use search bar that returns documents with a word “OKIC”. Issue is that it is not specified whether OKIC is in the document, as the document’s title etc.

**Download a document**

PDF downloading went without any problems

**Download an attachment inside the document**

Downloading an attachment inside a document went without a problem

**View an attachment inside the document**

Viewing an attachment inside a document went without a problem

**Add an event/group to a user**

It was hard to determine where to start with adding/deleting user.

**Go to Admin page**

Going to the admin page was simple, as there is a drop-down menu button with different positions.

**Delete previously create user**

Same experience as with viewing documents from the menu

**Edit access for that user**

No redirect after clicking button also not button to apply all. Other than that, the process when rather smooth.

**Log out**

The log out procedure worked as expected. No problems with it, was easy to find in the drop-down menu.

**Change your password**

Forgot password procedure was simple and easy to execute. Code came in fast and complying with the requirements e.g., with new password, was very easy.

**Additional comments both Positive & Negative**

Negative:

I can’t even see who I’m logged in as. User profile would be nice.

Menu button was not on all pages.

Some return buttons were very hard to find on non-existent.

Conclusion

We did the main changes the user has requested.