Test report for grade management system. Group 19

It speaks for itself we have to test whether the grade manager functions well. Despite the fact *Test Driven Development* (TDD) was recommended in the project manual, in the first instance we considered it to be good enough to either enter values into the database or see if the web application would display the correct values at the given time, or we would enter values into the web application and check in the database if these values appear in the database at the right location.

This worked well enough so far, and we were able to test our queries and ensure that they would return the right information or modify the right information, by simply making use of the functionality the query offers in the web application. If we would use the web application to add a `Student` object to the system, we would actually be testing whether we are able to test many things at once:

1. Whether the page is accessible.
2. Whether this page is displayed correctly.
3. Whether the information which the user entered is received.
4. Whether this information is handled.
5. Whether the database is updated correctly.

It was possible to do this because the project is on quite a small scale, additionally, it generally doesn't take a lot of time to use this kind of testing in a small project. Time definitely played a large role in our project, because it definitely limited the amount of functionalities we would've wanted to add.

The problem however is that in the case, that for example a page would not display, we would not know if the information which a user should've been able to enter on that page would be handled correctly. This means that if we want to test that, we have to ensure that the page functions properly before.

This is one of the reasons we are required to at least use unit tests as well. The advantage of those is that once you write them you will never have to think about them again, as they will run whenever the `maven` install/`maven` test is ran. They take some time to write, but in medium-sized projects they may well save a lot of time with testing. In the last week of the project we decided to implement some `JUnit` tests, we unfortunately had to keep most of the testing outside our scope, because we didn't have enough time.

Our `JUnit` tests cover the following:

Database	
Subject	Functionality of database tested
Student	Add/remove from database
Teacher	Add/remove from database
SuperCourse	Add/modify/remove
Course	Add/remove
SuperModule	add/modify/remove
Module	Add/remove

Security: We create an account with a certain password and generate a random salt / hashed password for this account. We then try a correct and a wrong password and see if it succeeds in checking the password.

Login: We create three different accounts, one for each of our roles. We then look if these accounts are correctly stored in the database, with their privileges correctly stored as well. Furthermore, we assert that passwords are correctly stored, and do not match nonsense passwords.

We also were introduced to Selenium, which is able to automatically test the entire web application. Unfortunately, we hard far too little time to even begin implementing this in our grade manager system.

Our main goal of making use of `JUnit` was not necessarily to test the application, for we already did a lot of testing on our own, as described above. It was just to show we have an understanding on how we should implement them and that we know they are very useful in testing. When we were testing the code, which we had tested on ourselves beforehand, with `JUnit`, we expected that we might've perhaps missed some mistakes, but we were glad to find out no test fails occurred.