Classic
Testing and Quality Assurance

by Team Classic

1 Introduction

In this document we will give an overview of the measures taken to ensure the quality of our project. The first part will discuss the testing strategy and guidelines and give an overview of all the testing that was done. The second part will handle how we assured the quality of our project. We will discuss our original point of view on testing and quality assurance and how this evolved during the project.

2 Testing

In the first weeks of this project a testing strategy and test plan were defined. This test strategy turned out to be a good original definition and only a few changes were made:

- We didn't succeed in running system tests automatically so we decided these would be run by the test expert every Sunday.
- We wanted to have a code coverage as high as 80%. This turned out to be quite difficult for our project because we have code we weren't able to test such as SQL exceptions in the database that will never be thrown. This is why the test coverage is the lowest in our persistence layer. We decided our new goal was 75%. Test results can be found in 2.3.
- We decided not to implement unit tests for the frontend. Frontend is only tested with the system tests in Selenium.

2.1 Test strategy and guidelines

This section describes our final test strategy and the guidelines that need to be followed to implement this test strategy.

2.1.1 Purpose

The purpose of a test strategy is to create a shared understanding of the overall approach, tools, targets and timing of test activities.

2.1.2 Guiding principles

- **Shared responsibility**: everyone is responsible for testing and quality. If you write a class you should also write test code for this class.
- **Test automation**: all types of tests (unit, integration, functional, acceptance) should be automated. Manual testing will only be used for usability testing.
- **Test management**: test cases, code, documents and data will be treated with the same importance as production code. This means that test code should also be documented.
- **Test independence**: at the beginning of each test as well as at the end the database should be empty. This ensures tests will not depend on each other and a test cannot cause failures for other tests.

2.1.3 Writing test code in Java

- Each test unit must be fully independent. Each of them must be able to run alone, and also within the test suite, regardless of the order they are called. The implication of this rule is that each test must be loaded with a fresh dataset and may have to do some cleanup afterwards. This is usually handled by setUp() and tearDown() methods.
- Names of functions in test code should clearly express: the method being tested, the expected input and the expected result. Variable names should express the expected input and state. Examples:
  - public void testSum_NegativeNumberAs1stParam_ExceptionThrown(int negative_number, int positive_number)
  - public void testSum_NegativeNumberAs2ndParam_ExceptionThrown(int positive_number, int negative_number)
  - public void testSum(int positive_number1, int positive_number2)
- We will use JUnit

Example:

```
import org.junit.Test;
import static org.junit.Assert.*;

public class TestSum {
    public void testSum(int positive_number1, int positive_number2){
        int sum = positive_number1 + positive_number2;
        assertEquals(sum, sum(positive_number1, positive_number2));
    }
}
```

2.1.4 Test types

Test	Description	Guidelines
Unit	Testing that verifies the implementation of software elements in isolation	Write unit test for all testable classes

Test	Description	Guidelines
Integration	Testing in which software elements, hardware elements, or both are combined and tested until the entire system has been integrated	Write integration tests for interaction between controllers and database, and entire interaction from API to database
Acceptance (System)	Testing based on acceptance criteria to enable the customer to determine whether or not to accept the system	Write Selenium tests for all acceptance tests

2.1.5 Test execution

Tests can be executed in Netbeans/Eclipse with JUnit. Please test the code before pushing to Git. Tests will also run automatically with Jenkins when pushed to the development branch. For some local tests a local database is needed. The section database in the readme of the projects contains instructions on how to set up a local database. Acceptance tests will not be run on the server but will be run manually every week by the test expert on Sunday. To run the acceptance tests we use the Selenium plugin for Firefox.

2.1.6 Quality and test objectives

The following quality attributes have been identified as relevant and are used as a basis for the test approach in terms of priority and test targets.

Attribute	Description	Measure and Target	Priority
Usability	The platform should be easy to work with	Real users should be asked about their experience Adding a cross-reference should be easy to add Asking a question should take at most 2 mouse clicks Adding an annotation should take at most 2 mouse clicks	Must have

Attribute	Description	Measure and Target	Priority
Correctness	Features and functions work as intended	100% completion of agreed features Severity 1 defects = 0 Severity 2 defects = 0 Severity 3 defects < 5 Severity 4 defects < 10	Must have
Maintainability	How easy it is to add features, correct defects or release changes to the system.	Code duplication < 5% Unit Test Coverage > 75% Method Length < 25 Lines	Must have

2.1.7 Classifying defects

Classification	Severity	Description
1	Critical	Defect causes critical loss of business functionality or a complete loss of service has occurred
2	Major	Defect causes major impact to business functionality and there is not an interim workaround available
3	Minor	Defect causes minor impact to business functionality and there is an interim workaround available
4	Trivial	Defect is cosmetic only and usability is not impacted

2.2 Test list

This section gives an overview of all the tests that have been written.

2.2.1 Unit tests

Unit tests for every class in backend, every developer is responsible for the unit test code of the code that he or she writes.

2.2.2 Integration tests

Test all API methods

All these tests are in /Backend/integtestsrc/endpoint/.

Name	Description	Input	Output	Location
testGetVideo	a video can be retreived	GET /course /1 /lecture /0 /video /1	json that contains name, youtube url and duration	Video EndpointIT
testGetVideos	all videos in a course can be retreived	GET /course /1 /lecture /0 /video	json containing a list of videos	Video EndpointIT
testPostVideo	a video can be added to an existing course	POST /course /1 /lecture /0 /video + json containing a name, youtube url and duration	json containing videoid	Video EndpointIT
testUpdateVideo	an existing video can be updated	PATCH /course /1 /lecture /0 /video /1 + json containing a name, url and duration	-	Video EndpointIT
testDeleteVideo	an existing video can be deleted	DELETE /course /1 /lecture /0 /video /1	-	Video EndpointIT
testVideo Not-FoundException	it is not possible to use a non-existing videoid	use non-existing videoid	4xx client error	Video EndpointIT
testVideo BadRequest	it is not possible to use a anything other than a valid json format	use a wrong json format	4xx client error	Video EndpointIT
testGet CourseNotes	course notes can be retreived	GET /course /1 /lecture /0 /coursenotes /1	json that contains name and url	CourseNotes EndpointIT
testGetAll CourseNotes	all coursenotes in a course can be retreived	GET /course /1 /lecture /0 /coursenotes	json containing a list of coursenotes	CourseNotes EndpointIT
testPost CourseNotes	course notes can be added to an existing course	POST /course /1 /lecture /0 /coursenotes + json containing a name and url	json containing coursenotesid	CourseNotes EndpointIT

Name	Description	Input	Output	Location
testUpdate CourseNotes	existing coursenotes can be updated	PATCH /course /1 /lecture /0 /coursenotes /1 + json containing a name and url	-	CourseNotes EndpointIT
testDelete CourseNotes	existing coursenotes can be deleted	DELETE /course /1 /lecture /0 /coursenotes /1	-	CourseNotes EndpointIT
testCourseNotes NotFoundException	it is not possible to use a non-existing coursenotesid	use non-existing coursenotesid	4xx client error	CourseNotes EndpointIT
testCourseNotes BadRequest	it is not possible to use a anything other than a valid json format	use a wrong json format	4xx client error	CourseNotes EndpointIT
testGet VideoComment	a comment can be retreived	GET /course /1 /lecture /0 /video /1 /comment /1	json that contains username, body, votes, approved, a list of replies and a list of selfreferences to videos	Comment EndpointIT
testGet CourseNotesComment	a comment can be retreived	GET /course /1 /lecture /0 /coursenotes /1 /comment /1	json that contains username, body, votes, approved, a list of replies and a list of selfreferences to coursenotes	Comment EndpointIT
testGet VideoComments	all comments of a video can be retreived	GET /course /1 /lecture /0 /video /1 /comment	json containing a list of comments	Comment EndpointIT
testGet CourseNotesComments	all comments of a coursenotes can be retreived	GET /course /1 /lecture /0 /video /1 /comment	json containing a list of comments	Comment EndpointIT
testPost VideoComment	a comment can be added to an existing video	POST /course /1 /lecture /0 /video /1 /comment + json that contains username, body, votes, approved, a list of replies and a list of selfreferences to videos	json containing commentid	Comment EndpointIT

Name	Description	Input	Output	Location
testPost CourseNotesComment	a comment can be added to existing coursenotes	POST /course /1 /lecture /0 /coursenotes /1 /comment + json that contains username, body, votes, approved, a list of replies and a list of selfreferences to coursenotes	json containing commentid	Comment EndpointIT
testUpdate VideoComment	an existing comment can be updated	PATCH /course /1 /lecture /0 /video /1 /comment /1 + json containing username and body	-	Comment EndpointIT
testUpdate CourseNotesComment	an existing comment can be updated	PATCH /course /1 /lecture /0 /coursenotes /1 /comment /1 + json containing username and body	-	Comment EndpointIT
testDelete VideoComment	an existing comment can be deleted	DELETE /course /1 /lecture /0 /video /1comment /1	-	Comment EndpointIT
testDeleteV CourseNotesComment	an existing comment can be deleted	DELETE /course /1 /lecture /0 /coursenotes /1comment /1	-	Comment EndpointIT
testComment NotFoundException	it is not possible to use a non-existing commentid	use non-existing commentid	4xx client error	Comment EndpointIT
testComment BadRequest	it is not possible to use a anything other than a valid json format	use a wrong json format	4xx client error	Comment EndpointIT
testComment UnauthorizedException	only the creator of the comment or an admin can manage the comment	use a non authorized user to perform an action	4xx clienterror	UnAuthorizedIT
testPostReply ToComment	a reply can be added to an existing comment	POST /course /1 /lecture /0 /comment /1 /children + json containing a body	json containing replyid	Reply EndpointIT

Name	Description	Input	Output	Location
testPostReply ToReply	a reply can be added to an existing reply	POST /course /1 /lecture /0 /reply /1 /children + json containing a body	json containing replyid	Reply EndpointIT
testUpdateReply	an existing reply can be updated	PATCH /course /1 /lecture /0 /reply /1 + json containing a body	-	Reply EndpointIT
testDeleteReply	an existing reply can be deleted	DELETE /course /1 /lecture /0 /reply /1	-	Reply EndpointIT
testReply Not-FoundException	it is not possible to use a non-existing replyid	use non-existing replyid	4xx client error	Reply EndpointIT
testReply BadRequest	it is not possible to use a anything other than a valid json format	use a wrong json format	4xx client error	Reply EndpointIT
testGet VideoReferences	all references friom a comment to a video can be retreived	GET /course /1 /lecture /0 /comment /1 /videoref	json containing a list of references	Reference EndpointIT
testGet CourseNotesRef-erences	all references from a comment to coursenotes can be retreived	GET /course /1 /lecture /0 /comment /1 /coursenotesref	json containing a list of references	Reference EndpointIT
testPost VideoReference	a video reference can be added to an existing comment	POST /course /1 /lecture /0 /comment /1 /videoref + json that contains videoid and timestamp	json containing referenceid	Reference EndpointIT
testPost CourseNotesRef-erence	a reference can be added to an existing comment	POST /course /1 /lecture /0 /coursenotes /1 /coursenotesref + json that contains coursenotesid, and a list of locations (pagenumber, x1,x2,y1,y2)	json containing referenceid	Reference EndpointIT
testUpdate VideoReference	an existing reference can be updated	PATCH /course /1 /lecture /0 /comment /1 /videoref /1 + json containing videoid and timestamp	-	Reference EndpointIT

Name	Description	Input	Output	Location
testUpdate CourseNotesReference	an existing reference can be updated	PATCH /course /1 /lecture /0 /comment /1 /coursenotesref /1 + json that contains coursenotesid, and a list of locations (pagenumber, x1,x2,y1,y2)	-	Reference EndpointIT
testDelete VideoReference	an existing reference can be deleted	DELETE /course /1 /lecture /0 /comment /1 /videoref /1	-	Reference EndpointIT
testDelete VCourseNotes-Reference	an existing reference can be deleted	DELETE /course /1 /lecture /0 /comment /1 /coursenotesref /1	-	Reference EndpointIT
testReference NotFoundException	it is not possible to use a non-existing referenceid	use non-existing referenceid	4xx client error	Reference EndpointIT
testReference BadRequest	it is not possible to use a anything other than a valid json format	use a wrong json format	4xx client error	Reference EndpointIT
testGetCourse	a course can be retreived	GET /course /1	json that contains name, lectures, videos and coursenotes	Course EndpointIT
testGetCourses	all courses can be retreived	GET /course	json containing a list of courses	Course EndpointIT
testPostCourse	a course can be added	POST /course + json containing a name	json containing courseid	Course EndpointIT
testUpdateCourse	an existing course can be updated	PATCH /course /1 + json containing a name, url and duration	-	Course EndpointIT
testDeleteCourse	an existing course can be deleted	DELETE /course /1	-	Course EndpointIT
testAuthorisation	a course can only be managed by a teacher or an admin	use a student to create a course	4xx clienterror	Course EndpointIT
testGetLecture	a lecture can be retreived	GET /course /1 /lecture /1	json that contains name, videos and coursenotes	Lecture EndpointIT

Name	Description	Input	Output	Location
testGetLectures	all courses in a lecture can be retreived	GET /course /1 /lecture	json containing a list of courses	Lecture EndpointIT
testPostLecture	a lecture can be added to an existing course	POST /lecture + json containing a name	json containing courseid	Lecture EndpointIT
testUpdateLecture	an existing lecture can be updated	PATCH /course /1 /lecture /1 + json containing a name, url and duration	-	Lecture EndpointIT
testDeleteLecture	an existing lecture can be deleted	DELETE /course /1 /lecture /1	-	Lecture EndpointIT
testGetUser	a user can be retreived	GET /user /1	json that contains name, and role	User EndpointIT
testGetUsers	all users can be retreived	GET /user	json containing a list of users	User EndpointIT
testPostUser	a user can be added	POST /user + json containing a name, a password and a role	json containing userid	User EndpointIT
testUpdateUser	an existing user can be updated	PATCH /user /1 + json containing a name, a password and a role	-	User EndpointIT
testDeleteUser	an existing user can be deleted	DELETE /user /1	-	User EndpointIT
Login test	an existing user can login	POST /login + json containing username and password	-	Login EndpointIT
credentials test	athe crednatials of the logged in user can be retreived	GET /login	json containing username and role	Login EndpointIT
Logout test	an existing user can logout	POST /logout	-	Login EndpointIT
LTI test	an tool consumer can generate a secret key that can be used to use our application	json containing secret key	new secret key to form secret key pair	Lti EndpointIT

Test all controller calls

All the controller calls that are used in the endpoints will also be tested in an integration test. This is the same test as the unit test, but it uses the real database instead of a mock-up.

2.2.3 System tests

All these tests are in /systemtests/.

Name	Description	Input	Output	File
Create account	User can create an account	username, password and role	you are logged in with your new account	create_account
Login	User can login	username and password	A user has access to all courses on the platform when subscribed. A user can subscribe to every course.	login
LTI	LTI Single sign on is supported: a user, logged in to another platform with LTI SSO support for Classic implemented, has access to the Classic web application without having to log in a second time.	Login on Moodle and launch the classic application	you are now logged in as a Moodle user and can use the application	lti
Create course	Admins and professors can create a course	name	a new course	add_course
Add lecture	Admins and professors can add a lecture	name	a new lecture	add_lecture
Add video	Admins and professors can add a video	youtube url	a new video	add_video
Add course notes	Admins and professors can add course notes	upload pdf	new course notes	add_coursenotes
Subscribe course	All users can subscribe to courses available on the platform	name od the existing course	When successfully subscribed, the course is visible on the user's home page	subscribe
Add comment	All subscribed users can add comments on a video at a certain timestamp or on coursenotes with a location	body and timestamp	the comment is visualized on the video	add_comment

Name	Description	Input	Output	File
Reply comment	Users can reply to comments on both videos and pdfs	body	the reply is visualized under the comment	add_reply
Add cross-reference	A cross-reference can be added, this is a link to course material. The linked course material can belong to the same lecture, another lecture or another course	select course material and a timestamp or pdf location	When clicking on the link yu go to the referenced course material	add_reference
hline Upvote a comment	All users can upvote or downvote all comments and replies	Upvote a comment	the vote has augmented with 1	upvote
Approve	Admins and professors can approve an answer	approve a comment	this will give a visual indication and mark the answer as correct	approve

2.3 Test results

2.3.1 Unit tests

The total coverage of the unit tests is 76%. The instructions that are not covered are mainly due to exceptions that can't be reached and we did not test functions that upload or download pdf's in unit tests.

Layer	Code coverage
API	82.4%
Controllers	79.3%
Models	100%
Persistence	71.2%

2.3.2 Integrations tests: Controllers - Database

The coverage of these integration tests is 81.3% in the controller layer. We did not test functions that upload or download pdf's in integration tests.

Layer	Code coverage
Controllers	81.3%
Models	71.7%
Persistence	48.8%

2.3.3 Integrations tests: API - Database

The coverage of these integration tests is 76.9% for all API endpoints. We did not test functions that download pdf's in integration tests.

Layer	Code coverage
API	75.9%
Controllers	71.4%
Models	74.1%
Persistence	39.2%

3 Quality assurance

Our main source of assuring quality in this project is the use of SonarQube. We made SonarQube run once a day to asses the quality of our code, and continuously used it to keep an eye on issues with the code, test coverage, documentation and code duplication. We always tried to fix important SonarQube issues as soon as possible.

Quality of the frontend was mainly tested by hands-on testing; both with Selenium and manually. We tried going through every possible scenario to ensure the behaviour of the web app is correct. This was done by various members of our team continuously over the course of the project; by the developer while developing and by another team member after a new feature was added or before a release.