TI3800 Bachelor's project - Shareworks

Final report

Projectcampus grading and review

Authors:
G.H. Kuijer
R. Nieuwenhuizen
M. de Vries

Supervisors:
M.A. Larson (Course supervisor)
Dr. T.B. Klos (TU Delft coach)
F. van Haaren (Shareworks)

November 25, 2013

Preface

This report describes, analyzes and evaluates the process around the design of the grading and review module and its development for the online learning environment Projectcampus. The module was developed as a bachelor's project. This project is the last part of the bachelor phase of the computer science study of the Delft University of Technology. The project entails developing a software product for an existing company while following normal software development procedures. The goals of this project are to teach students to work in a team of colleagues, in a real life company; to choose and follow a development method in cooperation with the client, and manage the products and processes that come with it; to determine quality requirements and assess them; and present and explain the results of the project. In this project, three students from the Delft University of Technology have developed a module for the existing Projectcampus system of Shareworks [1].

The project group wants to thank Shareworks for the opportunity of developing in their team and all the support they gave during the process of design and implementation. The group also wants to thank the mentor of the group Dr. T.B. Klos, who gave us great support in the form of reflection on the progress and reports.

Contents

Chapter 1

Introduction

Shareworks is a startup company which is developing an online learning environment [2] for project oriented education (see Figure 1.1). The name of this system is 'Projectcampus' [3]. It is a platform for schools and universities to facilitate project based education. The goal of the platform is to create an environment for students and teachers that supports easy information sharing. It allows students to easily share messages, files, videos and images. Other students or teachers can comment on them and express their approval. A team of students working together can start a project and share their posts on the project page. Any post can also be a submission for an assignment created by a teacher. For more information on Projectcampus, see chapter 2.

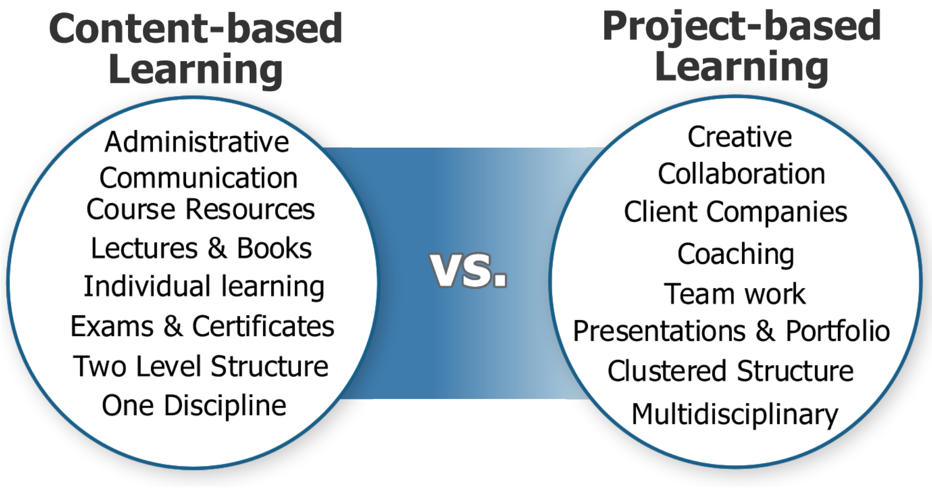


Figure 1.1: Content-based learning versus Project-based learning

One feature Projectcampus still lacks is the ability for teachers to grade the submissions from the students. The goal of this bachelor's project was to create a module that adds this feature, along with some related features. See chapter 3 for more information on the goals of the project.

The bachelor's project started with a research phase before proceeding to the design and implementation phases. The research phase was performed to answer the following question:

> What is the best solution to design and implement an online grading and review system within the Projectcampus system that is efficient, maintainable and user friendly?

The solution should be efficient so that the system can scale well when Projectcampus grows in popularity. It should be maintainable to make it easy to extend the system with new features and so Shareworks can modify the module easily if they want to. The solution should be user friendly so end users will be willing to work with the product and ensuring that the module attracts or keeps customers with Projectcampus. The result of the research phase is documented in Appendix B.

After the research phase the implementation started. The software development methods used during the development are explained in chapter 4. This includes a rough timeline of the work done during the project.

The design of the system is documented in chapter 5. This chapter explains what parts are found in the delivered module, how each part works and why it has been designed the way it is.

During all design and implementation phases of the project, the module has been tested to function correctly. How the module has been tested is explained in chapter 6. Besides automatic testing, the code has been reviewed by an independent third party: the Software Improvement Group [4]. The result of this review is documented in chapter 7.

The last two chapters of the report are the discussion and the recommendations found in respectively chapter 8 and chapter 9. These chapters build further on the knowledge acquired throughout the project. The discussion explains how the problem of the project has been solved and evaluates the process of the project, while the recommendations chapter does some further suggestions for Shareworks to improve the grading and review module and some other parts of Projectcampus.

Chapter 2

Projectcampus

Projectcampus is the software platform of the company Shareworks. The platform is designed to offer solutions to project based, collaborative education. It supplies a way of communicating within a project group and offers ways for a teacher to organise a course. This is important because the organisation of project courses requires a lot of energy, paper overhead and different levels of communication. Projectcampus facilitates everything from course communication to team collaboration and team coaching, and communication with client companies that gave an assignment. Additionally, it is important for the success of a course and its students that the organisation goes smooth and that students are stimulated and guided throughout the course duration. Projectcampus can help with this through its more engaging – compared to other education management tools suchs as Blackboard [5] – and attractive experience and functionalities. Innovative teachers now often turn to non-educational software products such as Facebook [6] to manage their courses while the use of such products for educational purposes is often not legal or gives rise to privacy issues. Further, research indicates that students like to separate their social network from their educational or career network [7].

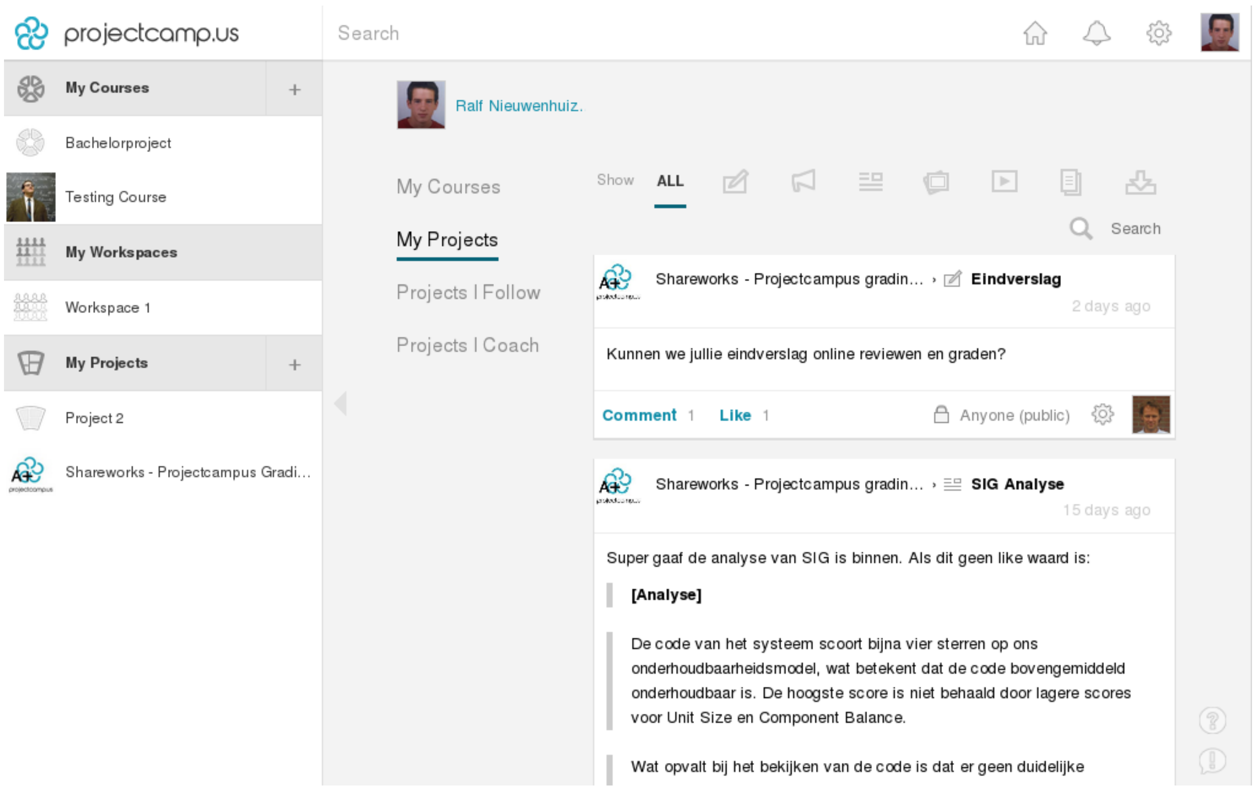
2.1 The structure

The structure of Projectcampus seems complicated, but once it is understood working with the software is easy. The group experienced this in the first week when they had to adjust to the software and had to get to know the structure. Especially the way in which workspaces fitted in courses and then projects fitted in either courses or workspaces was not clear at the first moment. In the system each user has the same rights, so each user is allowed to create a course, or an unbound project. A screenshot of Projectcampus can be found in Figure 2.1.

Creating a course creates a course page, on which the creator is given the 'staff' role. On this course page, teachers –who are given the 'staff' role– and students –with a 'student' role– can upload and share content with the participators of the course. A course reflects a course in real life.

On the course page, a teacher can choose to create workspaces. These are used when a course is divided into several topics. For example when three project groups are working on one topic and three other project groups on a different topic. A workspace can be a second level for a course, to separate these topics and create an organised environment. In real life, a workspace could be, for example, a separate classroom, with the exception that on Projectcampus workspaces can currently be managed by the students themselves. This can happen when a staff member that creates a workspace gives a student a 'manager' role in the workspace.

Within a course or workspace a project can be created. A project on Projectcampus is like a project group in real life. This again creates a page, which is now only meant for the project group itself, the course staff,



Shareworks - Projectcampus gradin... › Eindverslag
2 days ago

Kunnen we jullie eindverslag online reviewen en graden?

Shareworks - Projectcampus gradin... › SIG Analyse
15 days ago

Super gaaf de analyse van SIG is binnen. Als dit geen like waard is:

[Analyse]

De code van het systeem scoort bijna vier sterren op ons onderhoudbaarheidsmodel, wat betekent dat de code bovengemiddeld onderhoudbaar is. De hoogste score is niet behaald door lagere scores voor Unit Size en Component Balance.

Wat opvalt bij het bekijken van de code is dat er geen duidelijke

Figure 2.1: The projects feed of Projectcampus

and coaches. The project group members will be given the 'owner' role, because they own the project page. A coach is a person that can be invited by either students or teachers to coach the project. This can also be a person that is not a part of the course or organisation, creating the possibility for external companies that supply a project to coach the group working on it. The teachers of the coordinating course have the same rights as coaches, so they can view and supervise all project groups.

Projects can also be created outside the context of a course. This is useful for people who want to create a portfolio on Projectcampus without joining an educational institution, and enrolling for a course. These projects can for example be used for students who want to use Projectcampus for hobby or paid projects that they want to share. The 'coach' role gives a nice addition here, because two or more individuals can use the system without having to create a course. A coach can then be any user in the system not limited to the educational institution the users belong to. A figure of the structure of Projectcampus can be seen in Figure 2.2.

2.2 Possibilities

The Projectcampus software provides several possibilities. The main feature of the software is to give teachers and students the possibility to share information in an interactive way. Think of announcements, blog posts, images, videos, files, and assignments. How this sharing is designed is shown in Figure 2.3.

Assignments can be created in a course or in a workspace. In the course by users with the 'staff' role and in workspaces by users with the 'manager' role (further referenced to as 'staff'). While creating a new assignment the staff can set a deadline, set the expected type of submission (a file, an image, a video, or a

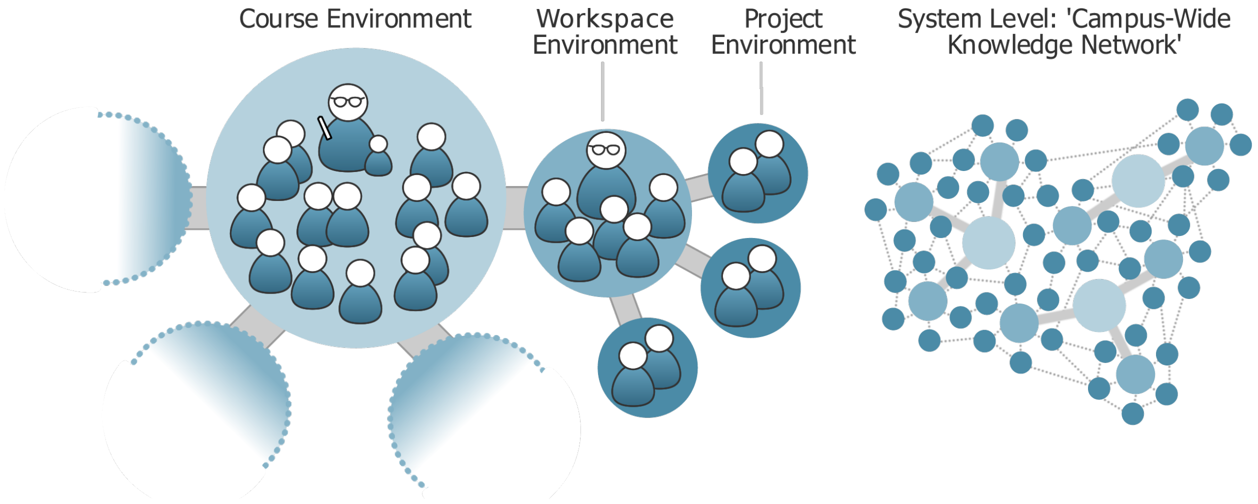


Figure 2.2: Structure of Projectcampus

blog), and choose whether the submission should be delivered by a project group or by an individual user. When the assignment is created students can submit their work on the assignment and the teacher can view these submissions. By supplying this feature Projectcampus makes it possible to hand in assignments in an easy way without wasting paper.

The goal of this bachelor's project is to extend this functionality by also giving grading and review functions to Projectcampus. Teachers are then able to review the documents online, making Projectcampus a more efficient way of working. Part of the administration process can then be done online.

2.3 The goal

The goal of Projectcampus is to improve collaborative learning, focussing on project based education, by providing a tool, ensuring that people do not have to use tools that are not primarily designed as online learning environment. In the long run the goal is to let academic institutions work with it worldwide.

Another goal of Projectcampus is to let students build a portfolio of projects that they have participated in. By doing this a student can easily show his skill set by referring to the projects in Projectcampus. This last goal is something that will only succeed if Projectcampus will grow in user count and in reputation.

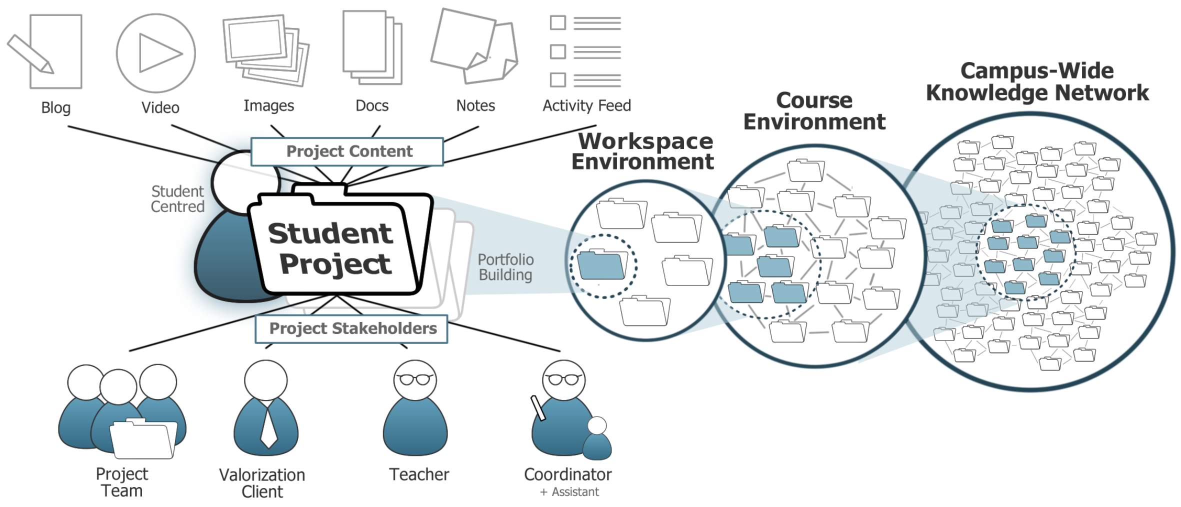


Figure 2.3: Sharing architecture of Projectcampus

Chapter 3

Project goal

This chapter describes the goal of this project and the problems that had to be solved, in section 3.1. Further it includes the list of requirements (section 3.2) and constraints (section 3.3) that had to be met in the project. It also elaborates on changes in these and why these changes were needed or thought to be useful. It also gives an overview in section 3.4 of how the user interface was proposed to be looking like after the project. Finally, there is a list of deliverables for this project in section 3.5

3.1 Goal summary

The goal of the project was to create a grading and review module in the Projectcampus system. This was one of the more important features still missing on the platform. The grading and review module should allow teachers and coaches to view submitted assignments, leave feedback on them and grade them.

The main problems in the original situation are:

- Teachers need a separate system for keeping grades.
- Teachers and students do not have an easy overview of the grades together with the submissions and assignments.
- There is no easy way of exchanging feedback between teachers and students.

To solve these problems a grading and review module came to mind. The module should implement ways of giving grades to submissions, give different overviews of grades to students and teachers and have options to give feedback on submissions.

3.2 Requirements

Several requirements were stated when the project was started. In the process some requirements have changed, some new requirements have been created and others have been removed. Below are the current requirements. The changes between these and the requirements in the project plan are discussed after the list.

1. The grading and review system must be a separate module that can be switched on and off.
2. The grading and review interface must be available through a review panel in the sidebar on the project page.

    (a) Project-based assignments must appear in this panel.

3. Teachers and coaches must be able to grade submitted work.
    (a) Both user-based and project-based assignments can be graded.
    (b) Different kinds of gradings can be given (dynamic, 1-10, pass/fail, A-F, +/-)
    (c) It must be possible to edit grades.
    (d) Grades will only be visible to students after publishing by the coach or teacher.
    (e) It must be configurable to publish grades delayed or instantly.
4. Teachers and coaches must be able to review submitted work.
    (a) Both user-based and project-based assignments can be reviewed.
    (b) It must be possible to edit reviews.
    (c) Reviews will only be visible to students after publishing by the coach or teacher.
    (d) It must be configurable to publish reviews delayed or instantly.
5. It must be possible to publish grades and reviews separately.
6. Students must be able to read reviews.
    (a) Students should be able to respond to given feedback.
7. Teachers and coaches must be able to share private notes with each other.
    (a) The notes must be stored on auto-save.
8. Students must be able to view all their grades in one overview.
    (a) Students must be able to search for specific courses and projects through their grades.
9. Teachers must be able to view all assignment submissions in one overview.
    (a) The overview must include grades and reviews when available.
    (b) The teacher must be able to view either user based assignments, project based assignments and both mixed in one view.
    (c) Project based assignments must be visibly distinct from user based assignments when mixed.
    (d) Project based results will count for all students in the project in the mixed view.
    (e) After clicking an assignment, a dynamic review panel for this user-based assignment must open.
        i. Switching between users with a forward/back button should be possible.
        ii. Switching between assignment with a forward/back button should be possible.
        iii. The forward/back buttons should react on the arrow keys on the keyboard.
10. Published grades should show up in the course activity feed.
11. All actions should provide visual feedback to inform the user of progress or failure.
12. Grades and reviews should generate notifications for students after publishing.
13. It should be possible to check submitted assignments for fraud.
14. When project group composition changes, existing grades should remain valid for the original users.

Changes in the requirements are not rare in a project [8] neither are they rare in this project. Below is the list of changes in the requirements that were made during the project and why they were made. See chapter 9 for recommendations that are left to Shareworks to improve the grading and review module after this project.

- Requirement 6a was added. The implementation could easily be done and it adds value to the system increasing the user-friendliness and efficiency.
- Requirement 9 has changed. Instead of all submissions, the grades of all submissions are shown on the grade page, it is more user-friendly as users will search for their grades on that page and increases efficiency of the system because every user does not have to click on the submission to see their grade.
- Requirement 9a has changed. Only grades are included in the overview but by clicking on the grade the feedback is shown, with a complete review panel. This is done because otherwise the page would be to full and users would not know where to click because of the many buttons available. By only displaying grades the user-friendliness has increased.
- Requirements 9b and c have changed. The filter between user-based and project assignments is not implemented, because the project assignments are already displayed per-user. Also filtering the assignments would create extra logic (means less maintainability) and a more complex page (less user friendly).
- Requirement 9eii was added. The implementation could easily be done and gives more efficiency to the system for the user increasing user-friendliness. As simple as this requirement is it will also not decrease maintainability of the system as not a lot of logic is needed. By this requirement teachers do not have to leave the submissions view anymore to view different submissions.
- Requirement 9eiii was added. This requirement was added because of user-friendliness reasons. Users might see it as intuitive to be able to navigate with the arrow keys. A design should be intuitive so by supporting this feature the user-friendliness will increase.
- Requirement 13 has been removed. It was too much work to implement. Further it was not entirely in the scope of the current project.

3.3 Constraints

During the process, both the group members and the client had to honour some constraints to keep the project going smoothly. The list of these constraints is given below. How these constraints have been met during the project is explained throughout the report.

1. The front-end and back-end must be separate systems.
2. The front-end and back-end must communicate using an API.
   (a) The API should be consistent with the existing Projectcampus API.
   (b) The API should be well documented. For each API call, the documentation should include:
      i. A description of what the API call does, including pre- and postconditions.
      ii. A description of every parameter passed to the API call, including its type and what it is used for.
      iii. A description of the returned value of the API call.
3. The front-end must provide a look and feel consistent with the rest of the Projectcampus front-end.
4. The front-end and back-end must be thoroughly tested.

(a) The back-end must have unit tests in Symfony.

(b) The front-end must be dynamically tested with automated tests.

5. The system must be secure against unauthorised access and modification.

3.4 User interface mock-up

The mock-up of the user interface was almost totally supplied by Shareworks, and was already displayed in the project plan (Appendix A). There was a good understanding of what functionality and panels were needed to create the grading and review module. The designer of Shareworks did give a good idea of how the panels in the front-end should look like. This section will describe the different panels and show some mock-up panels.

3.4.1 Review panel

The review panel was thought to be one of the most substantial parts of the grading system. It should pop up in various pages where the grading of submissions should be possible. The main page where it should be shown is the project page. It should allow students to view the submissions and the grades in a short time, for teachers and coaches it should supply a way to edit grades and give feedback. For the design of the review panel see Figure 3.1.

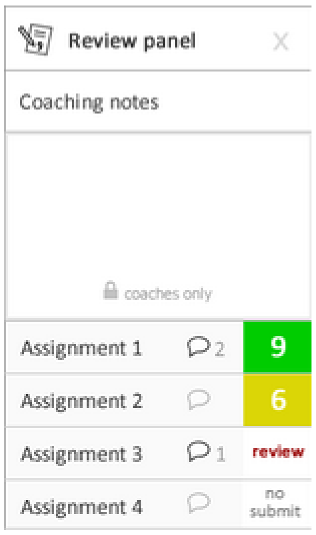


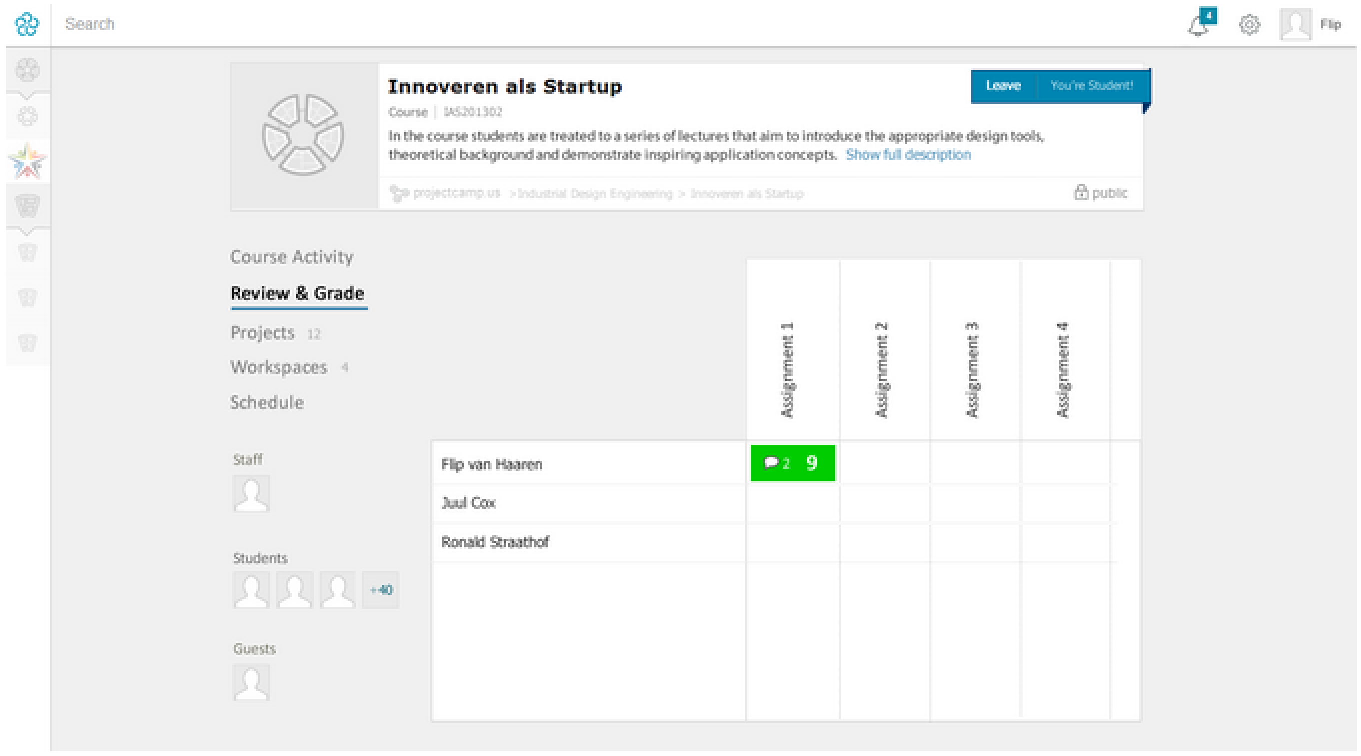
Figure 3.1: Review panel design

3.4.2 User grade page

Each student in the system should also be able to access his or her own grades page. In this page all the grades of the user are shown sorted by course. A search function should be available to filter the grades for specific projects or courses. A mock-up of this page is never created.

3.4.3 Course grade page

Teachers should have access to a page that lists all assignments and users of a course in one table. This table should include all the grades available but also display empty blocks where no submission or grade is available. The table has a cell for each submission that can be handed in. For a design of this page see Figure 3.2.



Innoveren als Startup

Course | IAS201302

In the course students are treated to a series of lectures that aim to introduce the appropriate design tools, theoretical background and demonstrate inspiring application concepts. Show full description

	Assignment 1	Assignment 2	Assignment 3	Assignment 4
Flip van Haaren	2 9			
Juul Cox				
Ronald Straathof				

Figure 3.2: Course grade page design

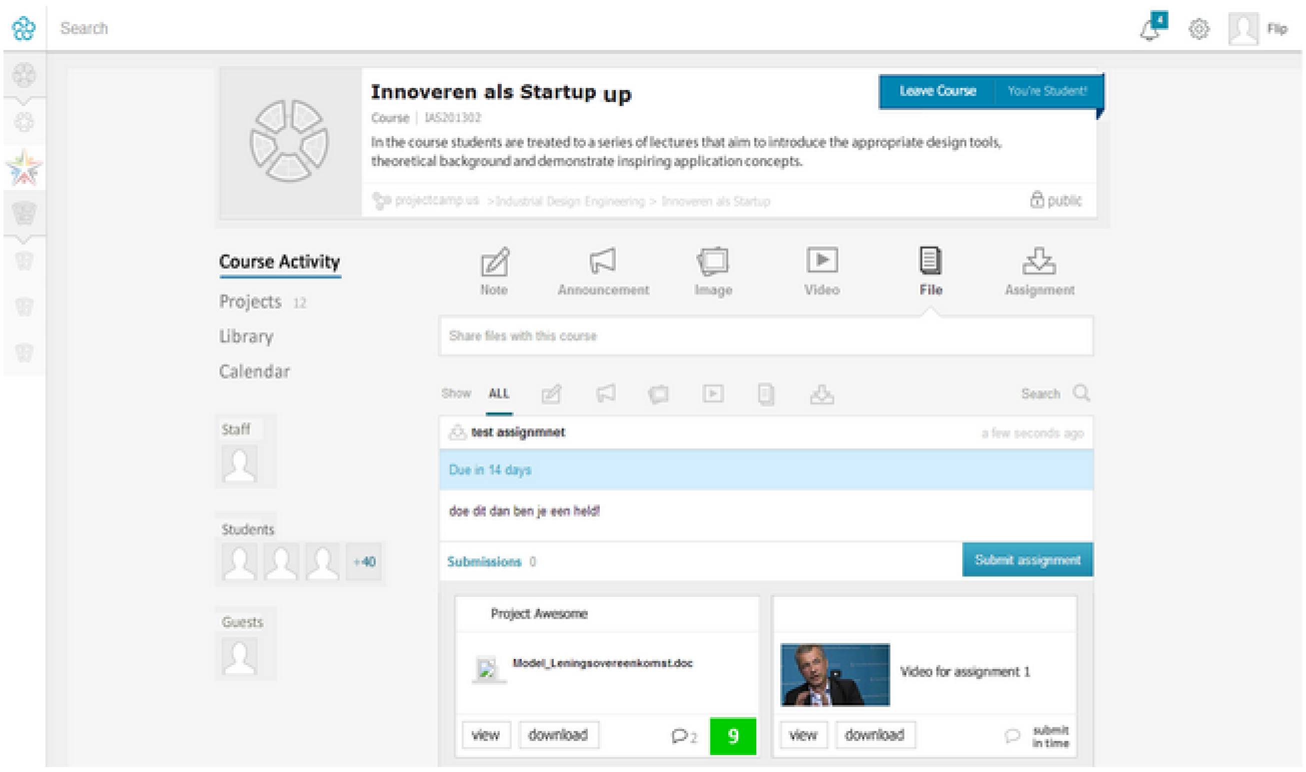
3.4.4 Course activity feed

The course activity feed already displayed all the items posted in the course. For the module a grade block had to be added as well as a feedback balloon. When someone clicks the grade block or the feedback balloon a new view should open with the submission displayed in it. Such a new view is called a modal. The modal was already used in the system to show a submission when the 'View' button is clicked. For a design of the course activity feed see Figure 3.3.

3.5 Deliverables

The deliverables of this project were:

- The code implementing the grading and review module;
- Testing code which tests the full implementation of the grading and review module;
- Documentation inside and outside the code on how the module works;



Innoveren als Startup up

Course | IAS201302

In the course students are treated to a series of lectures that aim to introduce the appropriate design tools, theoretical background and demonstrate inspiring application concepts.

projectcamp.us > Industrial Design Engineering > Innoveren als Startup

public

Share files with this course

test assignmnet — a few seconds ago

Due in 14 days

doe dit dan ben je een held!

Submissions 0

Project Awesome

Model_Leningsovereenkomst.doc

Video for assignment 1

Figure 3.3: Course activity feed design

- All reports written during the project, being the project plan, the orientation report, and the final report.

Chapter 4

Project methodology

This chapter describes the various methods that were used to complete the project. Further it gives the global planning that was made ahead of the project and the detailed planning that resulted out of the scrum planning for each week.

4.1 Project methods

For the bachelor's project, a phased development method [9] had to be followed (for example Scrum [10] or Behavior Driven Development [11]). The group has used the Scrum software development method for the project, because all members were familiar with working with Scrum, and Scrum served the need of being able to handle intermediate feedback given by Shareworks. Scrum is an agile programming method where the team works in iterations (called sprints) of one or two weeks. After each sprint, the team delivers a working product that has improved since the previous sprint. The project group has worked with sprints of one week, this was long enough to create a visible change in the product, and short enough to receive feedback from Shareworks in time, and plan adaptions in the next sprint. It is also good for testing purposes to have a working product each week (see chapter 6 for more information on how the project is tested).

All sprints started with an informal planning meeting where the team would decide what tasks would be worked on during the sprint. This weekly meeting mostly consisted of discussing which part of the system should be implemented that week. Exact tasks were not defined as the team was so small it could very effectively be split up while going on in the sprint. The planning of the sprints can be found in subsection 4.2.2. After each sprint, the team held a review meeting to see what was done and what can be improved for subsequent sprints. This meeting was mostly short and informal and was used to pinpoint specific trouble points such as which tickets had to be delayed and how troubling situations such as the merging of the code base of Shareworks into the grading and review code base should be solved.

Normally Scrum starts with small scrum meetings each day. This was not done in this project. The project team was so small that everything was constantly discussed while working. Normally when the project team is bigger such meetings are necessary. It is impossible to keep track of 5 or 6 other persons working at the same project without having meetings. When developing with just three persons it is possible to keep track of this while working, though three persons seems to be the limit. In groups of 4 pair programming will soon occur, creating the situation that one pair does not exactly know what the other pair is doing because they are too busy with their own task. With just three persons problems are easier discussed with all group members during programming.

The team has also employed test-driven development. This forced the team members to think how to implement a feature first, write a test for the feature and finally implement the feature. This provided an easy way to confirm that features are working as they should and prevents tests being added as an afterthought. Although test-driven development was not strictly used for each feature, it was used for creating the object model (see section 5.2). The appointment to never commit anything that was not tested was held. This still created the situation that the group members had to write tests and think about their implemented features.

4.2 Planning

For this project a rough planning was made in the first weeks. The rough planning included when which iteration was planned but the tasks of what to do in the iteration was only planned in the scrum meetings. The planning made in the scrum meetings resulted in the iteration planning.

4.2.1 Global planning

Below is the global planning made in the first weeks of the project. The final presentation is moved to week 49. This was the best moment for all the attendees.

Weeknumber	Global task	Partial tasks
38	Orientation	Compile project proposal
		Create project plan
		Compile project requirements
39	Orientation	Create orientation report
40	Work	First iteration
41	Work	Second iteration
42	Work	Third iteration
43	Holiday	
44	Work	Fourth iteration
		Send code to SIG
45	Work	Fifth iteration
46	Work	Sixth iteration
		Create final report
47	Work	Complete and send final report
		Send code to SIG
48	Work	Final presentation

4.2.2 Iteration planning

The global planning above suggests there are six iterations in the project. There was a rough outline on the order of the bigger tasks, and based on that, each week the group decided what tasks would be worked on. During iteration 6 an extra sprint was planned for week 47. This sprint was reserved to complete the last tasks, clean up the code and deliver the code to Shareworks.

Iteration 1

- Setup environments
- Create testing environment
- Test existing code

Iteration 2

- Design object model
- Implement object model
- Create API calls for grades and feedback
- Create grade block directive
- Add grade block to item tiles
- Add feedback tab and feedback balloon to item tile

Iteration 3

- Implement user grade page
- Implement course grade page
- Implement API for grade scales
- Implement assignment weights

Iteration 4

- Create functionality to add grade scale to course and assignment
- Calculate scores in grade block with grade scale
- Calculate averages using grade scales and assignment weight
- Clean code written for SIG
- Implement review panel on project page
- Add publishing of grades and feedback on course grade page

Iteration 5

- Create service to check permissions in front-end
- Create administration page
- Create grade scale center
- Refactor code with recommendations from SIG
- Split all grading and review functionality into grading bundle
- Add browsing through submissions in modal
- Move average and grade level calculation to back-end

Iteration 6

- Merge Shareworks code base with grading and review code base
- Add ‘SerializationGroups’ to API calls
- Refactor API doc to be consistent with Shareworks docs
- Add form validation in grade scale center

Iteration 7

- Add option to switch off module in front-end
- Integrate publishing of feedback and grades with notification system
- Clean up code
- Merge grading and review code base with Shareworks code base

Chapter 5

System design

This chapter describes the different parts of the grading and review module. It documents design decisions made during the project and explains why the eventual implementation has deviated from the initial design at some points.

This chapter first explains how the system is split in a front-end system for the user interface and a back-end system that manages the data (section 5.1). Next, section 5.2 describes the object model used by the system and how it changed during the course of the project. The API is documented in section 5.3 while section 5.4 details the design of the front-end components.

5.1 Global system layout

This section describes how the Projectcampus system is split in a front-end and a back-end. It will explain what the tasks and responsibilities are of both the front-end and the back-end.

Initially, the back-end and front-end were part of the same repository, where most of the front-end files (but not all) were part of the 'web' directory. This original combined repository was not structured very clearly, so during the project Shareworks decided to split it into separate repositories for the front-end and the back-end.

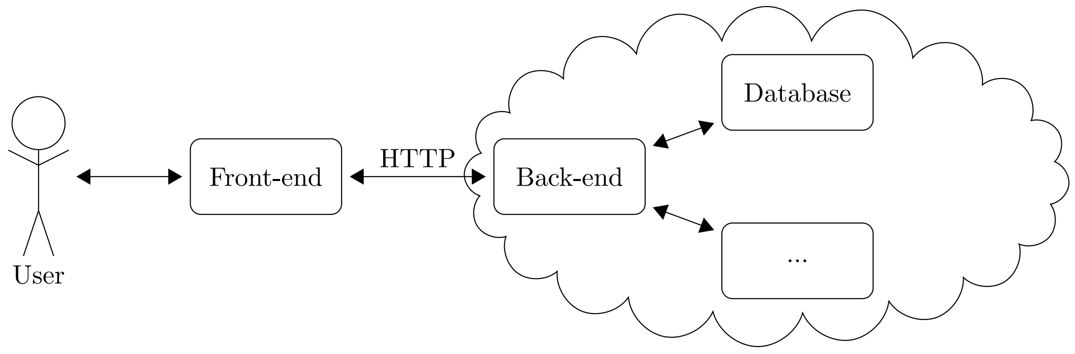


Figure 5.1: Global system layout of Projectcampus. Components inside the cloud run on servers on the Internet.

5.1.1 Front-end

The front-end is a separate web application written in Javascript, HTML and LESS [12]. It is a user interface for Projectcampus and it does not directly communicate with a database. Instead, the front-end sends requests to the back-end over HTTP and the back-end decides what to do with the request. See Figure 5.1 for a schematic overview.

While the front-end is hosted on the Internet, it is primarily a Javascript application which means that all code runs within the browser of the users. This is why Figure 5.1 depicts the front-end outside of the cloud. It also brings security concerns with it, but those can be remedied by ensuring a user has the proper authorisation in the back-end as well as in the front-end.

The grading and review module adds new pages and interface elements and contains some changes to existing front-end components. More details can be found in section 5.4.

5.1.2 Back-end

In contrast to the front-end, the back-end is written in PHP and runs on a server. This means the users have no access to the code and cannot bypass security checks by simply modifying the client-side code.

When the back-end receives a request, it will first check if the requesting user has the proper permissions. If not, the request is denied. Otherwise the request is fulfilled and the back-end will retrieve or modify data from the database, depending on the exact request.

The API exposed by the back-end follows the REST architectural style. This should ensure an easy to use the API by exposing supported operations on different types of resources in a consistent manner [13]. In essence, the back-end functions as a gatekeeper to the database and some other services, preventing unauthorised retrieval or modification of data, and limiting the allowed operations on the data. See Figure 5.1 for a schematic overview.

5.2 Object model

The object model describes what kind (or classes) of objects exist within the system, and what properties these objects have. These objects are used to represent important entities in the system, such as the users, courses, projects and grades. In line with modern software engineering methods, the object model was designed at the start of the project and updated during each iteration when necessary [9].

See Figure 5.2 for a diagram of the entities in the original Projectcampus system that are relevant for the grading and review module. Note that this diagram is somewhat simplified as it does not list all properties of the objects, but only the more important ones.

The grading and review module adds more entities. As described in chapter 3, the module needs to support grading and reviewing submissions. This means two new entities: grades and feedback. Additionally, the module should support multiple grade scales, which also need to be added to the model. Finally, average grades turned out to have a slightly different interface from regular grades, so they are represented by different classes in the model (see subsection 5.2.2 for more details). See Figure 5.3 for a simplified overview of the changes made to the object model. The remainder of this section describes the changes to the object model in more detail.

5.2.1 Grades

The Grade class represents the grade for a submission. The grades are linked to a submission and a user.

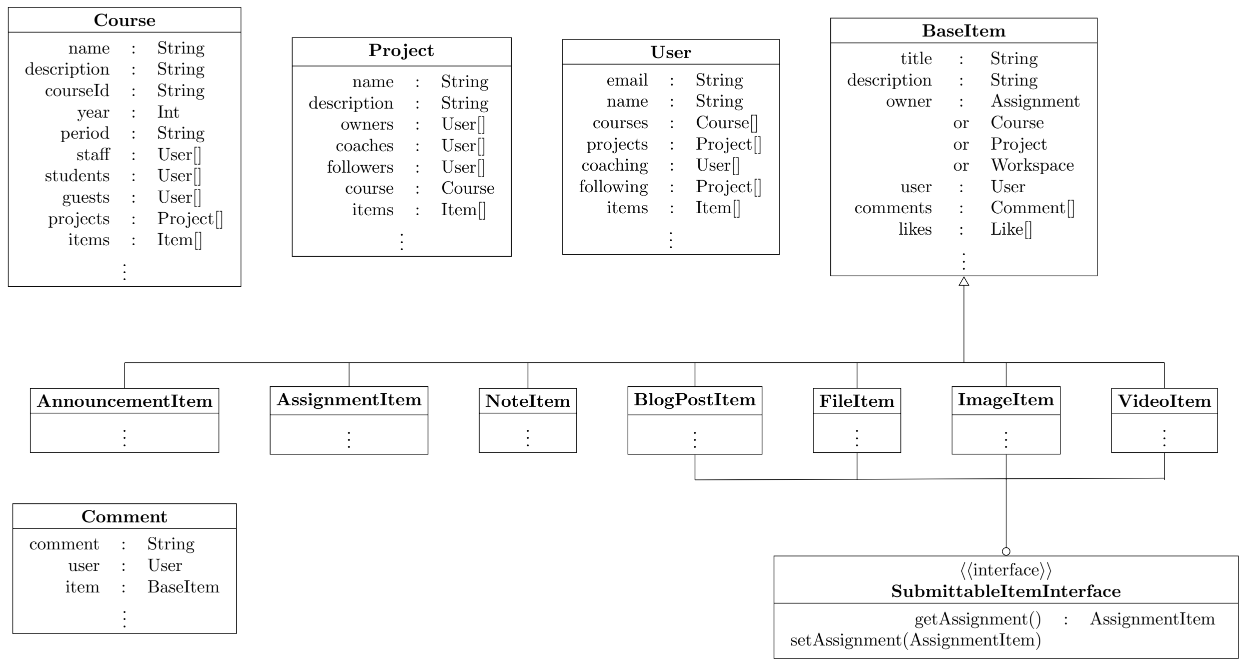


Figure 5.2: Simplified object model of the original Projectcampus system. Not all classes and properties are shown.

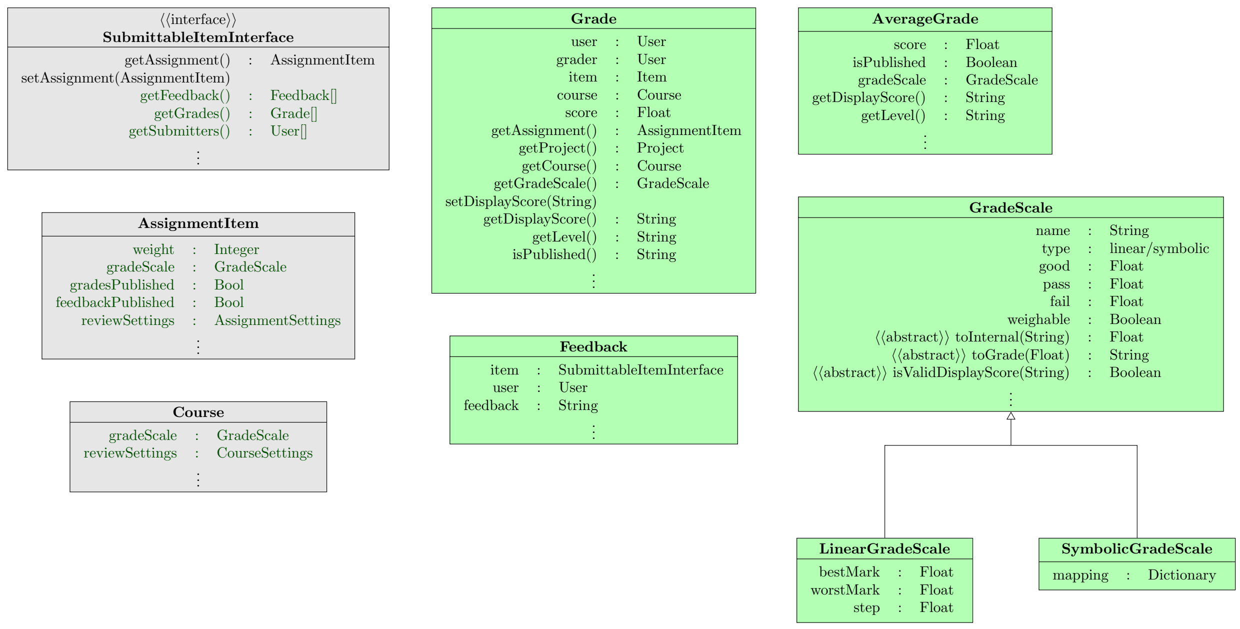


Figure 5.3: Simplified object model of the original Projectcampus system. Not all classes and properties are shown.

The score is stored as a float that can range from 0 to 100, inclusive. This score can be used to calculate the average of multiple grades, but it is not intended to be shown to users. It should be translated to or from a human readable score by a grade scale first.

As section 3.2 states, grades should remain valid for the original submitters. Without this requirement, it would be sufficient to store a single grade with a submission. However, when the project group composition would change, the grade would suddenly count for the new group members rather than for the original members. This would happen because the only way to see who the grade is for would be to check who the project members are of the project owning the submission.

To fix this, every user gets a grade for a submission. An alternative would be to store the original submitters with a submission. Both solutions ensure that grades remain valid for the correct user. However, storing grades per user also allows for individual grading. The added benefit of individual grading is why the first solution was chosen.

Grade objects also store a link to the course of the assignment. This information is already available by looking up the submission with the grade, the assignment with the submission and finally the course with the assignment. However, the database system used by Projectcampus does not allow searching through links [14]. To allow retrieving all grades for a course anyway, the grades are linked directly to a course. Without this link, all course submissions would have to be retrieved just to get all grades in the course. Letting the database perform the search means that less data is transferred between the database and the back-end and provides better performance.

Data duplication like this does pose problems and should be avoided in general [15]. Besides the storage overhead, it means that when the data changes all occurrences have to be updated. In this case however, the course of an assignment – and with it the course of a grade – is not allowed to change. This means that there is no need to ensure data integrity after the initial creating of a grade, and the storage overhead is acceptable.

Data integrity still has to be ensured during the creating of a grade. To achieve this, the interface of grade objects does not allow the programmer to set the course manually. Instead, the course is linked automatically when a submission is linked. This prevents the programmer from making a mistake by linking the wrong course or no course at all, which makes the system more maintainable.

Earlier versions of the Grade class included more duplicate information. Grades were also linked directly to an assignment and a project to allow searching on those fields. Eventually it became apparent that it is actually never required to search for grades based on an assignment or project. Instead, the submissions for an assignment or project are retrieved and the grades along with them. At this point the redundant properties were removed, although the Grade class still has convenience methods to retrieve the removed properties. These getters use the linked submission to find the proper assignment and project.

An additional convenience getter was added to get the grade scale used by the assignment. The grade scale is then used by get/setDisplayGrade() to translate between internal scores and human readable scores.

5.2.2 AverageGrades

Early versions of the grading and review module used the same Grade class to represent both grades and averages. The only difference was that the averages were never stored in the database, but calculated when needed.

In contrast to regular grades, average grades might not actually have a link to a submission. While a project average for a submission does, the average of a user for a course is not linked to a single submission. Using the original Grade class for averages was not appropriate and required weird workarounds to link the correct grade scale with an average. A new class was added for average grades, which is not linked to a submission. Instead it allows the programmer to manually set the grade scale used.

This allowed the Grade class to have stricter requirements, because it does not also have to be used for averages. For example, it is now required for a grade to be linked to a submission. This in turn means the grade must also always be linked to an assignment and course, although indirectly through the submission. Setting stricter requirements on the Grade class can prevent accidental mistakes and improves the maintainability of the system.

5.2.3 Feedback

Feedback objects store a user, a message and some metadata such as when the feedback was posted and last modified. It is very similar to the Comment class which also stores a user, a message and metadata. In early versions of the module the Feedback class was even a subclass of the Comment class. This meant that feedback was always compatible with comments, and the front-end also used the same templates to display the two. This improved maintainability, since any changes to comments are automatically reflected for feedback, even in the front-end. However, it caused trouble with the security system.

The security system uses voters to determine if a user is authorised to perform a given action on a given object. Normally only one voter votes on one object so there is exactly one vote. However, since feedback subclassed comments, every feedback object was also a comment object. That meant that the comment voter also voted on feedback, and the comment voter was too lenient in allowing permission. Everyone who can view an item can view the comments, but only the submitters, coaches and staff are allowed to view the feedback on a submission. In this case, the two conflicting votes were resolved by granting permission when just one of the voters granted permission.

The security issues here outweigh having to maintain two separate classes. To remedy this issue, the Feedback class was separated from the Comment class, although it is still very similar.

A third option would have been to add a base class for both and put the shared members there. This would have solved the security issues while keeping the benefit of low maintenance costs. It would have been the preferred option, except that the message field of a comment is called ‘comment’. When it became clear Feedback could not inherit from Comment, the field was also renamed. Ideally the field would have a more generic name like ‘message’ for both the Feedback and the Comment class, but the Comment class is not part of the grading and review module and could not be modified for the module.

5.2.4 GradeScales

Grade scale objects provide a way to translate between human readable grade scores and internal scores. For example, a grade scale might use characters from the alphabet to represent grades. However, calculating the average of A, C and F is impossible without assigning a numerical value to the characters. Even worse, the symbols do not necessarily form a linear scale. The difference between A and B might be larger than between B and C.

The grade scale objects define how the human readable scores are converted to a numerical value. The GradeScale class does not store this mapping itself, but defines three abstract methods. One method to check if a string can be converted to an internal score, one to do the actual conversion, and one to do the conversion the other way. The actual implementation of these methods is delegated to two subclasses: LinearGradeScale and SymbolicGradeScale.

The abstract base class allows sharing common code between linear and symbolic grade scales while still providing specialised behavior where necessary. This means code is not duplicated across the two grade scales and improves maintainability.

Instead of storing grade scale objects in the database, a small number of built-in grade scales could have been added. However, this would require hard-coding the available grade scales somewhere in the system and would make it difficult to add new grade scales in the future. This would not be very maintainable, so instead the grade scales are stored in the database and can be managed from the front-end.

LinearGradeScale

The LinearGradeScale class can be used for simple numerical, linear scales. The only properties that need to be defined are the best mark, the worst mark and the step size between them. A valid score is any that is between the worst and best mark, inclusive, and that can be reached from the worst mark by increasing or decreasing it by a multiple of the step size.

The best mark can be lower than the worst mark. When this is the case, a lower mark is better than a higher mark and will be translated to a higher internal score.

SymbolicGradeScale

Symbolic grade scales can be used in all cases where a linear grade scale is not appropriate. However, it requires an explicit mapping of all valid symbols to an internal score. Additionally, the mapping must be bijective so that every allowed symbol can be translated to an internal score unambiguously, and vice versa.

Defining a grade scale of this type takes more effort, so in cases where a LinearGradeScale can be used that is often preferable.

5.2.5 SubmittableItemInterface

SubmittableItemInterface is an interface that already existed in the original Projectcampus system. It has been extended to add getters to retrieve grades, feedback and the users responsible for the submission. Who exactly these users are depends on what type of assignment the submission is for. For individual assignments, only the user who submitted the assignment is responsible and only that user should receive a grade. For project based submissions, all the students in the project group are responsible.

Additionally, a trait was added that implements the interface. This trait can be used on items instead of implementing the interface for each item type separately. Changes to the interface now have to be implemented in only one place instead of four, which improves maintainability.

5.2.6 AssignmentItem

The AssignmentItem class already existed in the original Projectcampus system. It has been extended to include a link to a grade scale, an assignment weight and two booleans to remember if grades and feedback have been published.

The assignment weight is used to calculate the average grade of a user for an entire course. The linked grade scale is used for all grades for the assignment submissions.

Review settings have also been added to allow users to choose whether or not to use the grading and review module for a specific assignment. Grades and feedback can be turned off separately to allow users to use one feature without the other. This is more user friendly than forcing users to use both features just to use one of the two.

5.2.7 Course

The Course class also existed in the original system. Like assignment items, it has been extended to have a link to a grade scale. The grade scale is used for the average grade of a student in the course, and it is used as default grade scale for all new assignments in the course.

Review settings have also been added to allow users to choose whether or not to use the grading and review module for a specific course. Grades and feedback can be turned off separately just as with assignments. When the setting is turned off for the course, it cannot be turned on for an assignment.

5.3 Application Programming Interface

The back-end system implements an API that the front-end can use to retrieve, store and modify data from the database. Calls to the API are made by sending HTTP request with the proper URL and optionally a request body containing additional data [13].

This section describes the additions to the back-end API for the grading and review module. The final API has diverged from the original design. Where applicable, the changes from the original design are explained.

5.3.1 Grades

For user-friendliness, the API allows for retrieving, editing and deleting grades for all users for a submission, and for a single user for a submission. This way the coach or teacher can just as easily grade students individually or all together. When retrieving all grades for a submission, an average is also included.

These API calls changed since the original design. Initially, actions on individual grades used the ID of the grade. However, since a user can have at most one grade for a submission, the API was changed to identify single grades based on a submission and a user.

This means that it is not required to know the grade ID in order to change it, and it means that the URL of a grade is always known, even when there the grade does not exist yet. So instead of a POST method to store a grade and a PUT method to update it, a single PUT method to store or update the grade is enough. This means there is one less API call to maintain.

/items/{itemId}/grades/

GET	Get the grades for all users for a submission
POST	Set the grade for all users for a submission
DELETE	Delete the grades for all users for a submission

/items/{itemId}/users/{userSlug}/grade/

GET	Get the grade for a single user for a submission
PUT	Set the grade for a single user for a submission
DELETE	Delete the grade for a single user for a submission

5.3.2 Project submissions

One of the requirements was to add a review panel to the project page that shows the grades for all submissions in the project (see section 3.2). Teachers and coaches should also be able to grade the submissions. To do so, a new API call was needed that retrieved all those submissions.

This API call was not part of the original API design. Instead, the API included a call to retrieve all grades for a project. However, grades are returned along with submissions, and the review panel needs to show all submissions anyway. So instead of retrieving the submissions and grades separately, this call returns both at the same time.

This means the front-end does not have to retrieve the submissions and grades separately, and then execute a loop to find the correct grades with the submissions. This is more efficient and easier to maintain.

/projects/{slug}/submissions/

GET	Get all submissions for a project

5.3.3 User grades

The grading and review module must also be able to present a student with all their grades in one overview. One API call is provided to retrieve the relevant information for this overview. This includes the average for each course.

/users/{slug}/grades/

GET	Get all grades for a user, indexed on course id

5.3.4 Course grades

The grading and review module also contains a course grade overview which is similar to the user grade overview. However, this overview is for teachers and shows the grades for all users for all assignments in the course. An API call is available to retrieve all these grades.

/courses/{slug}/grades/

GET	Get all grades for a course, indexed by assignment ID

5.3.5 Feedback

The feedback API is very simple. It allows to retrieve all feedback for a submission, to post new feedback and to update and delete feedback. The original design also allowed retrieving one feedback message based on its ID. However, a single feedback message lacks the context of previous feedback messages, so there is no sensible use case for this API call. It has been removed from the API, removing the burden to maintain this API call when the system changes.

/items/{itemId}/feedback/

GET	Get all the feedback belonging to an item
POST	Post feedback for an item
PUT	Edit a specific feedback item
DELETE	Delete a specific feedback item

5.3.6 Assignments

As stated in the requirements (section 3.2), feedback and grades should not be visible to students immediately. Instead, the feedback and grades should remain hidden from students until the teacher publishes them. This allows teachers to grade all students before publishing them in one go.

Two API calls are added to allow for this. One call to publish the feedback, and one to publish the grades. This is done per assignment and not per course so that it is possible to publish the feedback and grades of different assignments at different times.

/items/{itemId}/publishFeedback/

PATCH	Publish feedback of an item

/items/{itemId}/publishGrades/

PATCH	Publish grades of an item

5.3.7 Grade scales

The API to interact with grade scales is simple. It is possible to retrieve all grade scales, or only one based on an ID. Additionally, grade scales can be edited and deleted one at a time. These API calls did not change since the original design, although the URL has changed slightly due to the way an internal library works.

The edit and delete calls will refuse to modify a grade scale that is already in use. As explained in subsection 5.2.1, grades are stored with an internal score. To translate that score into a human readable grade, the corresponding grade scale is used. If the grade scale changes the displayed grade also changes, even though the grade itself was not edited. This is not very user friendly. Even worse, if the grade scale is deleted it becomes impossible to display the grades to a user. To prevent this from happening, it is simply not allowed to modify or delete grade scales that are in use.

/grade/scales/

GET	Get all grade scales
POST	Create a new grade scale

/grade/scales/{scaleId}/

GET	Get a specific grade scale
PUT	Edit a specific grade scale
DELETE	Delete a specific grade scale

5.4 Front-end

The front-end is the system that is running in the browser of the user. This system communicates with the back-end by generating calls to the API. Because the front-end had to display all the new content that could be generated by the back-end, new pages, directives, filters, and services had to be created. All these will be described in this section. Explanations about what directives, filters and services are will follow in the next subsections.

5.4.1 Pages

The grading and review module added interface options to existing pages, but some new pages were added as well.

Course grade page

The course grade page displays all the grades in the current course. To meet constraint 3[1] the course grade page is hung into the grade page. This makes sure that the already existing layout is used. The grades are efficiently displayed in a table with students and assignments, as can be seen in Figure 5.4. Both for assignments and for users, an average grade is shown. By clicking the grade blocks (section 5.4.2), an item modal (section 5.4.2) is opened, where feedback can be posted, and grades can be given. Both per submission as per user. When a grade is changed in the item modal, and the modal is closed, a message will be shown, asking the user to refresh the page, because averages may have changed. These averages need to be recalculated in the back-end. By calculating this in the average constraint 1[1] is met keeping logic in the back-end. The user-friendliness of this is achieved by keeping the table compact and by this making sure the user keeps the overview of the grades.

There is also a section with an overview of which grades and feedback have already been published to students and a column telling how many students have already submitted the assignment. When the reviews have not been published, students cannot see them. By clicking the provided buttons, teachers can publish grades and/or feedback for the desired assignment.

This page can be accessed via the course page, and is only visible to staff, making the grades secure against unauthorised access and meeting constraint 5[1].

User grade page

Users will want to see the results of their hard work in one overview, for that purpose, the user grade page was designed. In the user section, there is a 'My Grades'-button that brings the user to this page. The user will see an expandable list of all the courses he or she has ever received grades for, with an average for each course. A search bar is also available to search through the list by course name, assignment name, submission name or project name. By clicking on a specific course, a list of all assignments for that course with their individual grades will be shown. Clicking on any of these grades will open the item modal, where the submission will be loaded, and eventual feedback can be read. A view on this page can be seen in Figure 5.5. The user-friendliness in this view is achieved by creating a search bar to search through the grades and also by collapsing and expanding course sections to present the grades. These sections makes sure the user is not flooded with data.

Administration page / Grade scale center

Because grade scales can now only be managed by administrators, and Projectcampus did not have a specific administration page yet, this page was created. The administration page style is copied from an already existing page so constraint 3[1] is met. For now, the only option on the administration page is to manage grade scales, but more options will probably be added in the future.

In the grade scale center, new grade scales can be created, and existing grade scales can be edited or deleted (as long as they are not in use yet). On this page custom form validation takes care of feedback to the user, creating a more user-friendly experience. An overview of this page can be found in Figure 5.6.

[1]section 3.3
[1]section 3.3

5.4.2 Directives

In AngularJS, a directive is a small, re-usable piece with underlying functionality. By creating these maintainability of the system will increase. For the grading and review module several directives have been created, which are described below.

Grade block

The grade block is the most used directive that is created for the grading and review module. The grade block gets input several input parameters, and based on those it creates a grade block which has different functionalities. All the possible forms of the grade block are shown in Figure 5.7. A grade block always has a grade parameter. From this, it gets the score to display. An optional array of grades can be given, which generates the popover that is shown on Figure 5.8. When a grade block is editable (also a parameter) a submission is needed, to be able to change the grade, and save it to the database. The user the grade belongs to is also required in that case. Two function parameters can be given, a function to execute when the grade block is clicked on, and a function to execute when the grade is successfully saved. The last optional parameter tells the grade block whether there is a submission at all to be graded.

When the grade is edited a form appears to fill in the grade. For symbolic grades there are only a limited amount of options. Therefore it is user-friendly to have a drop down menu with options. Although there were issues implementing this cross-browser this feature is implemented. For linear grade scales the validation of the number entered is custom written. Some browsers did not support the validation that is standard included in html, this is why custom validation was needed.

Feedback balloon

The feedback balloon is currently only placed in the course feed, on a small item tile, as can be seen in Figure 5.9. The feedback balloon is exactly what it says, a small text balloon that on mouseover shows the first feedback that was given to the submission. Next to the balloon there is a number, showing the number of feedback for this item. When there is no feedback, the balloon is greyed out. By clicking on the balloon, the item modal is opened. More on the item modal can be found in section 5.4.2.

Grade panel

The grade panel is a panel that displays a list of users with their corresponding grade for a certain submission. It is used in the review panel (section 5.4.2) and in the item modal (section 5.4.2). The grade panel was first developed as part of the review panel, but it was later split into a separate directive so it could be re-used in the item modal enhancing maintainability.

Review panel

The review panel is a directive that shows all submissions in a project along with the grades for those submissions. It is only visible on a project page, and only for project owners, project coaches or course staff. The project owners see only one grade for each submission, which is their own grade. Coaches and staff see the grades for every project owner and an average per submission. They can also give grades using the review panel. See Figure 5.10 for a screenshot of the review panel.

Early versions of the review panel were responsible for creating the entire list of submissions and users, but this functionality has been moved to the grade panel that is also used in the item modal. The final version of the review panel simply uses one grade panel per submission.

Item modal

A modal is a streamlined, but flexible, dialog prompt with the minimum required functionality and smart defaults [16]. The item modal was already used to view a submission in the original system, and the grading and review module added feedback and grades to it, as can be seen in Figure 5.11. Because of all the added functionality, the modal was transferred to a directive. On the course grade page, when the modal is opened, teachers can use arrow buttons to switch between users and submissions. This improves the workflow and with that user-friendliness, because they do not have to close and open a new modal for grading every user. The arrow keys on the keyboard can also be used for this purpose. In the course overview, this feature can also be used, but now only for switching between submissions.

Focus me / Select me

The Focus me and Select me directives can be placed on an input field. When the input field is shown, the keyboard focus will be on this item, and the text in this input field will be selected. This is used for editing grades in grade blocks. This directive also decreases the amount of clicks needed for editing which enhances user-friendliness.

5.4.3 Filters

In AngularJS, filters are functions that convert a text string to a certain format. An example of an already existing filter in Projectcampus was TimeAgo, that created a string '... X days ago' from a date. Filters can be re-used in different pages enhancing maintainability.

Truncate

Truncating text is a well-known phenomenon on websites. When text gets too long, it should be cut off somewhere, and the missing text should be replaced by a placeholder, mostly three dots ('...'). For this purpose, a re-usable filter was created, that does exactly this, and it has a configurable maximum length and placeholder.

ConcatUserGrades

When hovering over an average grade block, a popover will become visible, where all the owners of this submission are listed, followed by their grade Figure 5.8. To format this data the right way, the ConcatUserGrades filter was created.

5.4.4 Services

In AngularJS, services are the communicators in the front-end with the API in the back-end. For the grading and review module, four new services were created.

Feedback

The Feedback service ensures that feedback can be collected from the database, and that it is saved on the right place. Another function of this service is to send the publish request to the back-end.

Grades

The Grades service ensures that grades can be collected from the database, and that they are updated upon change. This service is divided in two parts. One for grades and one for user grades. The splitting of this service is done to make more clear that there is a difference. In the back-end these calls have also been split into two controllers. In the end these splits make the classes and services shorter and clearer increasing maintainability.

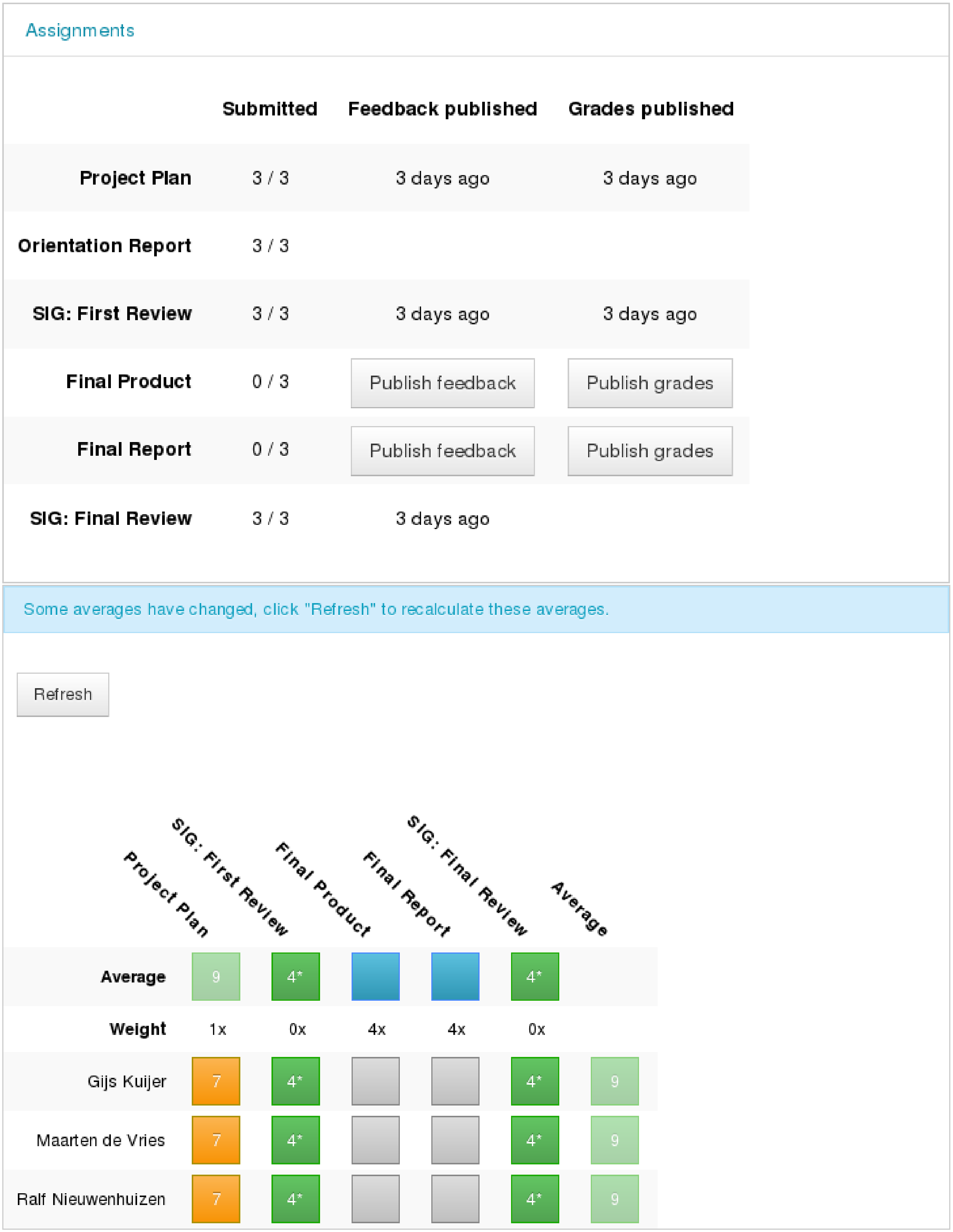
GradeScales

The GradeScales service handles all the requests about grade scales, and is therefore only used by site administrators.

Permissions

The Permissions service is created to reduce code duplication in the front-end. In the front-end the permissions where mostly done in the pages. It could happen that one method was duplicated three or four times. Because of this the project group saw the necessity of a service. This service has functions to check if a user has a certain role, or if a user can edit a certain grade, for example. This service is highly recommended to be used by Shareworks in the future, and to be extended with even more permission checks. With this service constraint 5[1] is also more likely to be met as all the permissions checks are on one place. If the same permission check is repeated on several places it is more likely that one of them has an error in it. If only one check exists and is used it makes sure that if there is an error it will be visible on every spot it is used. Decreasing the code duplication of permissions also made it more easy to maintain the system, making bug fixes easier.

[1]section 3.3



Assignments

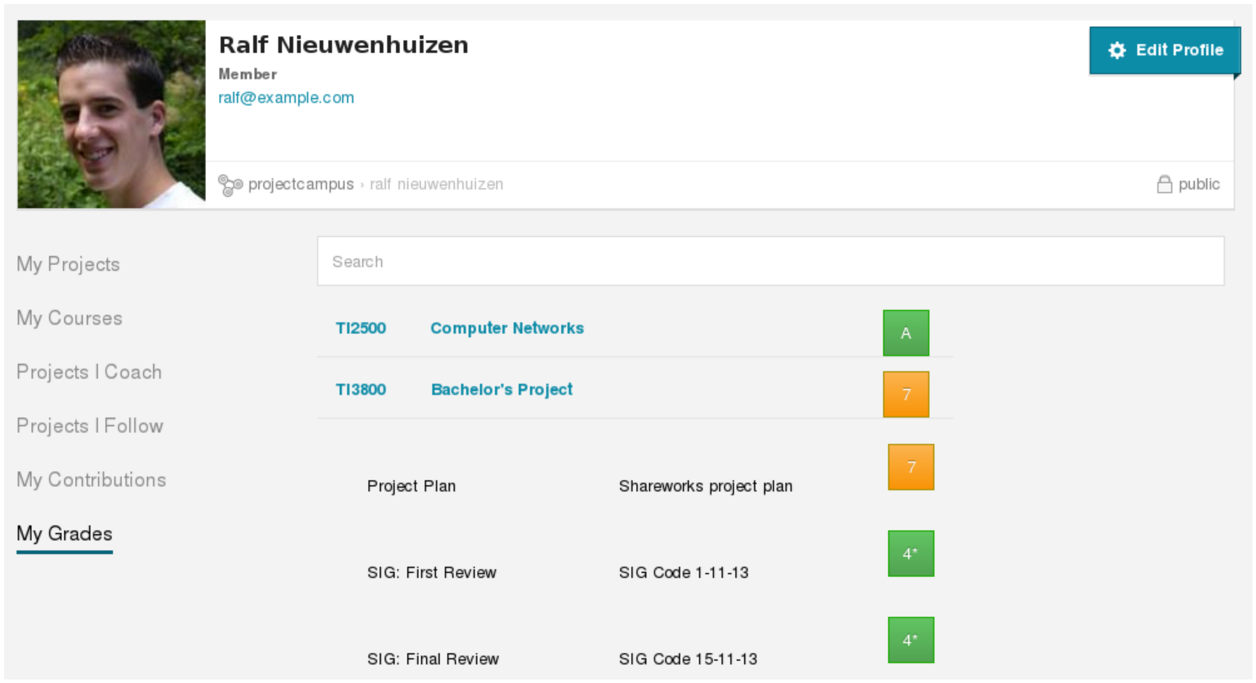
	Submitted	Feedback published	Grades published
Project Plan	3 / 3	3 days ago	3 days ago
Orientation Report	3 / 3		
SIG: First Review	3 / 3	3 days ago	3 days ago
Final Product	0 / 3	Publish feedback	Publish grades
Final Report	0 / 3	Publish feedback	Publish grades
SIG: Final Review	3 / 3	3 days ago	

Some averages have changed, click "Refresh" to recalculate these averages.

Refresh

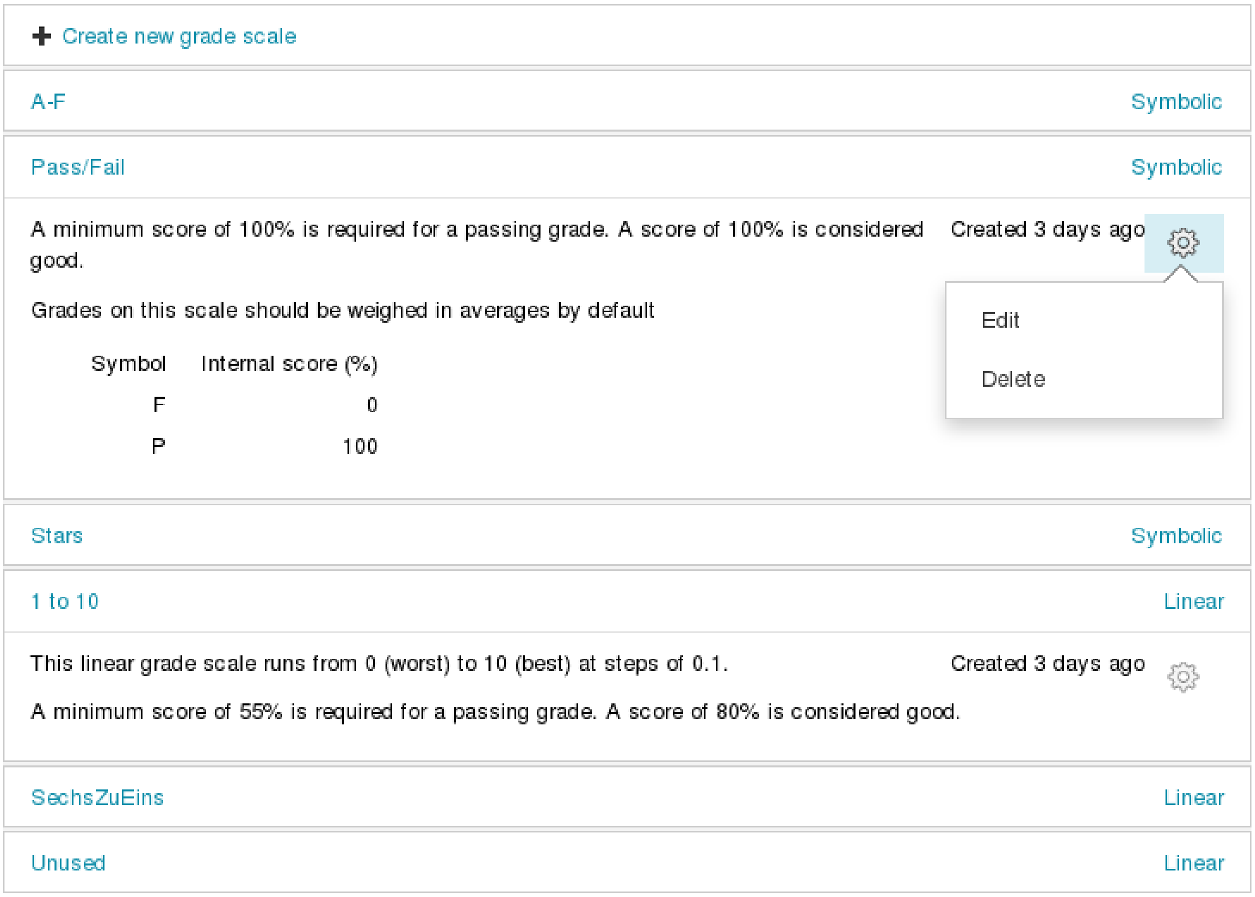
	Project Plan	SIG: First Review	Final Product	Final Report	SIG: Final Review	Average
Average	9	4*			4*	
Weight	1x	0x	4x	4x	0x	
Gijs Kuijer	7	4*			4*	9
Maarten de Vries	7	4*			4*	9
Ralf Nieuwenhuizen	7	4*			4*	9

Figure 5.4: The course grade page.



TI2500	Computer Networks	A
TI3800	Bachelor's Project	7
Project Plan	Shareworks project plan	7
SIG: First Review	SIG Code 1-11-13	4*
SIG: Final Review	SIG Code 15-11-13	4*

Figure 5.5: The user grade page.



Create new grade scale

A-F Symbolic

Pass/Fail Symbolic

A minimum score of 100% is required for a passing grade. A score of 100% is considered good.

Grades on this scale should be weighed in averages by default

Symbol	Internal score (%)
F	0
P	100

Created 3 days ago

Stars Symbolic

1 to 10 Linear

This linear grade scale runs from 0 (worst) to 10 (best) at steps of 0.1.

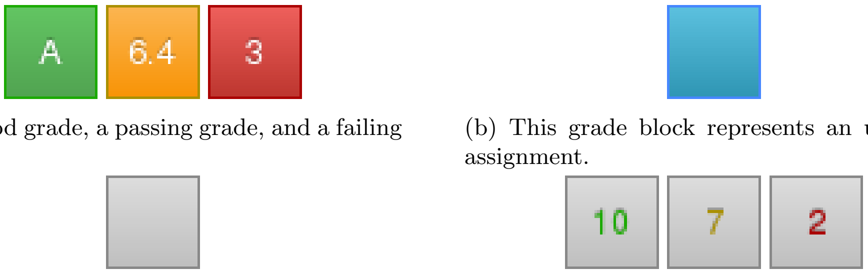
A minimum score of 55% is required for a passing grade. A score of 80% is considered good.

Created 3 days ago

SechsZuEins Linear

Unused Linear

Figure 5.6: The grade scale center on the administration page.



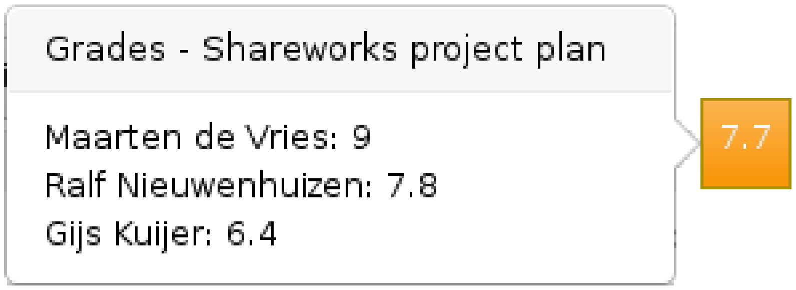
(a) A good grade, a passing grade, and a failing grade.

(b) This grade block represents an ungraded assignment.

(c) This grade block represents an empty spot, meaning nothing is submitted yet.

(d) A good grade, a passing grade, and a failing grade, all not yet published to students.

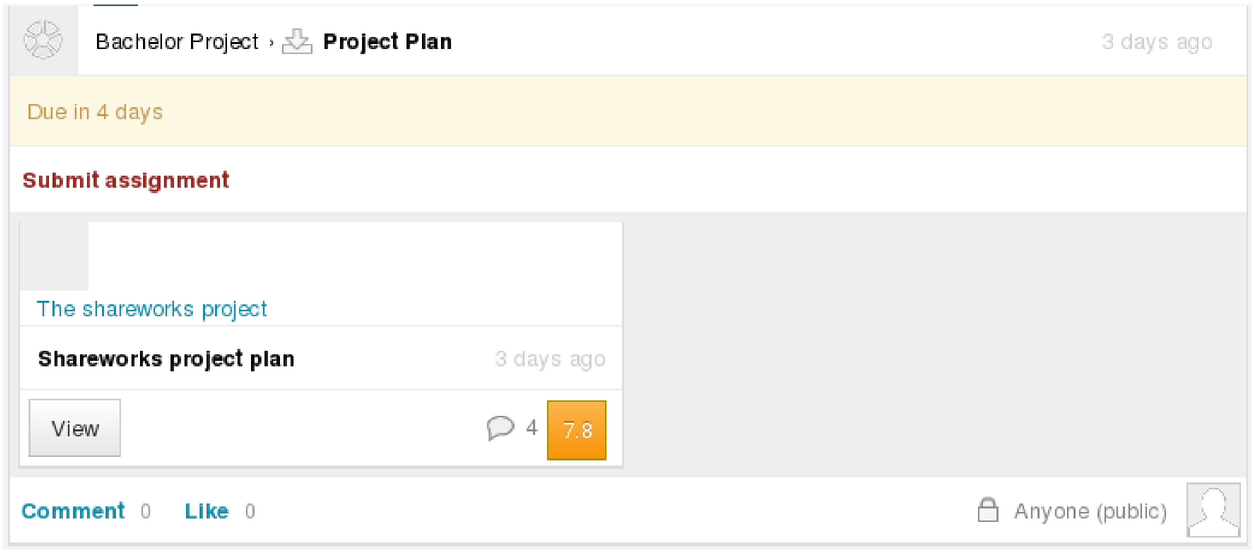
Figure 5.7: The different types of grade blocks.



Grades - Shareworks project plan

Maarten de Vries: 9
Ralf Nieuwenhuizen: 7.8
Gijs Kuijer: 6.4

Figure 5.8: Popover for an average grade.



Bachelor Project › Project Plan

3 days ago

Due in 4 days

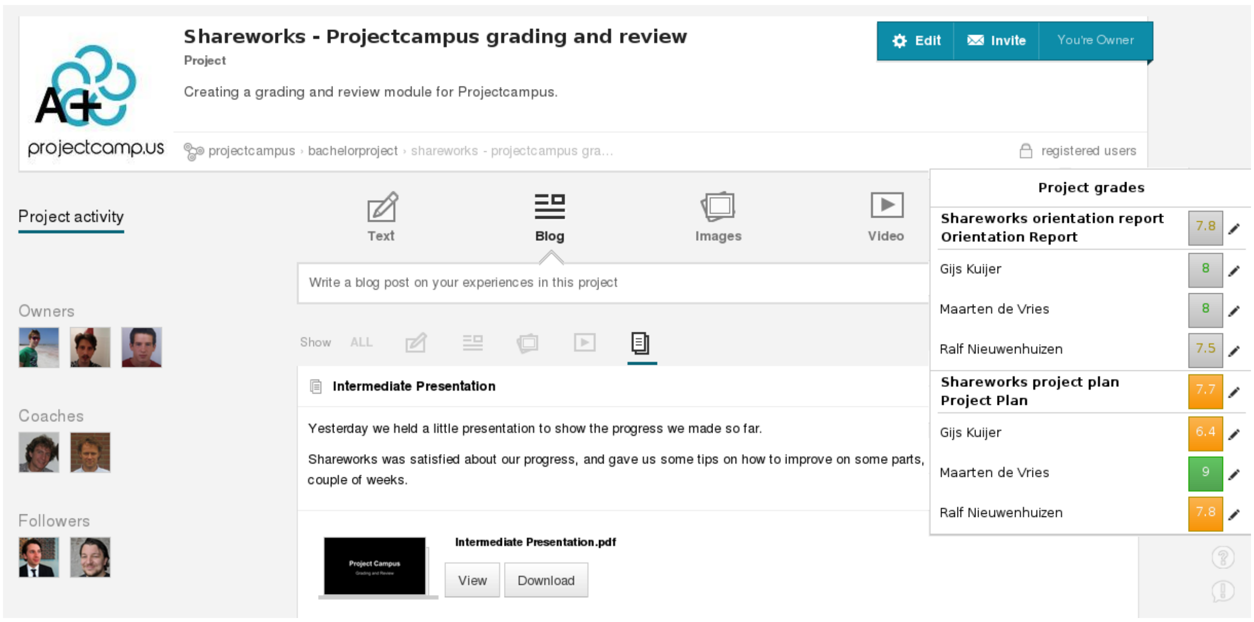
Submit assignment

The shareworks project

Shareworks project plan

3 days ago

Figure 5.9: The feedback balloon in the course overview section.



Shareworks - Projectcampus grading and review

Project

Creating a grading and review module for Projectcampus.

projectcampus › bachelorproject › shareworks - projectcampus gra...

registered users

Write a blog post on your experiences in this project

Intermediate Presentation

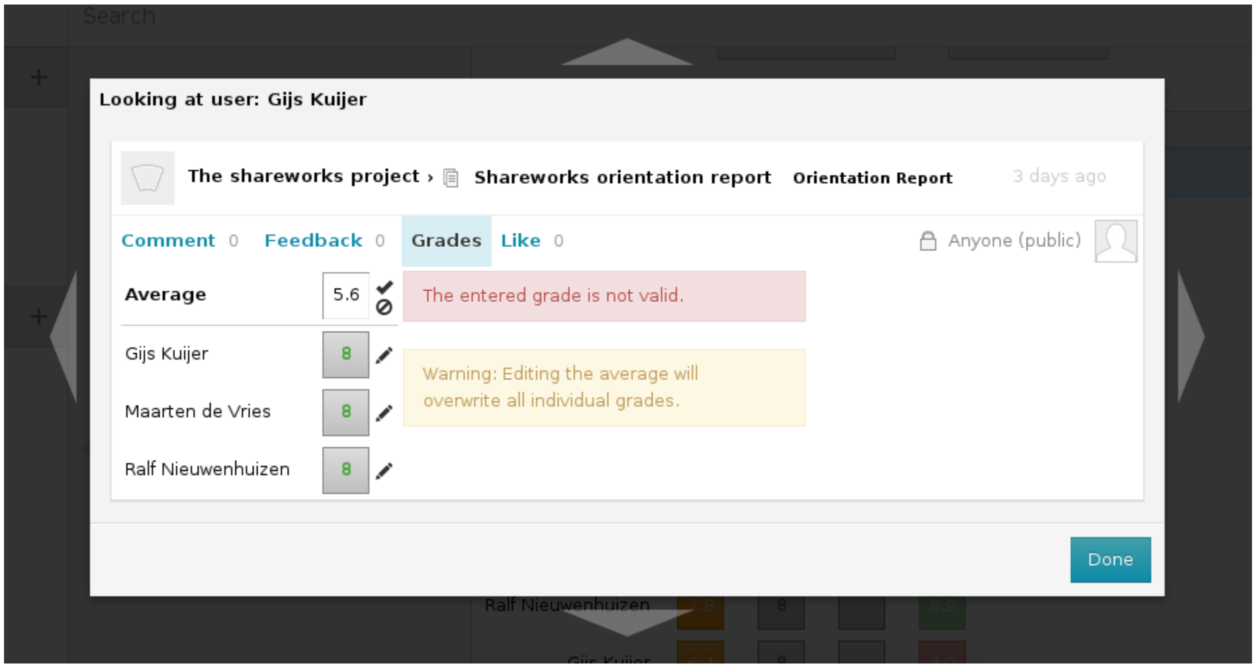
Yesterday we held a little presentation to show the progress we made so far.

Shareworks was satisfied about our progress, and gave us some tips on how to improve on some parts, couple of weeks.

Intermediate Presentation.pdf

Project grades	
Shareworks orientation report Orientation Report	7.8
Gijs Kuijer	8
Maarten de Vries	8
Ralf Nieuwenhuizen	7.5
Shareworks project plan Project Plan	7.7
Gijs Kuijer	6.4
Maarten de Vries	9
Ralf Nieuwenhuizen	7.8

Figure 5.10: The review panel on the project page.



Looking at user: Gijs Kuijer

The shareworks project › Shareworks orientation report Orientation Report 3 days ago

Comment 0 Feedback 0 Grades Like 0 Anyone (public)

Average	5.6
Gijs Kuijer	8
Maarten de Vries	8
Ralf Nieuwenhuizen	8

The entered grade is not valid.

Warning: Editing the average will overwrite all individual grades.

Figure 5.11: The item modal.

Chapter 6

Testing

In this project, Test-Driven Development, **TDD**, is used. In TDD a test suite is designed and written, and executed with failures as the functions that are tested are not implemented yet. Then, the function under test is implemented, and corrected until the test passes. If the test suite was written correctly, the code is working correctly as well.

In practice, TDD was most useful for creating the new voters deciding over the permissions in the Project-campus back-end. TDD forces you to think over all the possibilities in advance, and that revealed all the paths that the voter had to be able to take. TDD was also good for creating new Documents. These were created from scratch, and also they form the entire model, on which the rest of the system is built. Therefore they had to be thought over thoroughly.

6.1 Unit tests

Unit testing is the testing of small parts of software. In the early implementation phase, unit tests are written, and only after that, the functions under test are written. This way, design is thought over before the code is implemented, which makes you rethink some of your design. This has proven to be useful, and made sure parts of the design were refactored. This happened when design was not thought over enough and cases were forgotten while designing. Combined with proper documentation about the expected value and why that value is expected, the unit tests provide an additional consistent, always up-to-date documentation. When changing a function, you are always warned by unit tests if the result also changed. If it has changed, you should check if you really want to make the test result change, or if you made a mistake rewriting the function. This has really proven to be useful, because running the unit test was always a nice way to find out why things did not work as expected.

In the resulting project, there are unit tests for all Documents in the GradingBundle, as well as for all Managers (interactors with the database), and Voters (permission checks for user actions). This is for the back-end, all these tests can be run using PHPUnit. The tests for the Documents assert that the validators work correctly, so corrupted data cannot be entered into the database, and they test complex functions, for example the conversion methods in the GradeScales, from the displayed score to the internal score. The tests for the Managers assert that data is sent to the database correctly, and that is returned the same way it was sent. The tests for the Voters assert that authorised users can perform their actions, and unauthorised users cannot.

In the front-end, there are very few unit tests. Front-end testing is done via Karma, which has a syntax that is very clear to read, and useful for documentation. There were some new filters that had to be created for this project, and for these filters, unit tests were written to assert they work well in all cases. Apart from these filter tests, there are no automated unit tests in the front-end, because front-end testing is usually done via functional tests only.

6.2 Functional tests

Functional tests, or integration tests, assure that software components work together well. They assure that all function calls are transported correctly, and that the program works over a wider spectrum, in contrast to the isolated scope of unit tests.

In the resulting project, there are functional tests for all Controllers in the GradingBundle. These tests, again, confirm that authorised users can perform their actions, and unauthorised people cannot, but they also test if the desired result of performing these actions is achieved.

Functional testing in the front-end consisted solely of user tests. The programmer that was writing the code asserted that the desired result was achieved, and that there were no unwanted results. After commiting and pushing the source code to the repository, the other programmers tested the feature as well, and made sure it was working correctly. During these user tests and even during some demos for the client several bugs were still found. This is also some form of user acceptance testing. Although there are no automated functional tests, no harm can be done to the system, because the back-end is properly tested and unauthorised actions cannot be done.

As the front-end does not have functional tests constraint 4b[1] is not met. There was decided not to meet this constraint because the front-end was under constant change. This resulted in many updates in tests after each change in the front-end. Having tests for the heavy changing front-end costed too much time, while clicking around in Projectcampus did reveal most of the bugs. This is only most of the bugs because it is not possible to say if all the bugs are revealed.

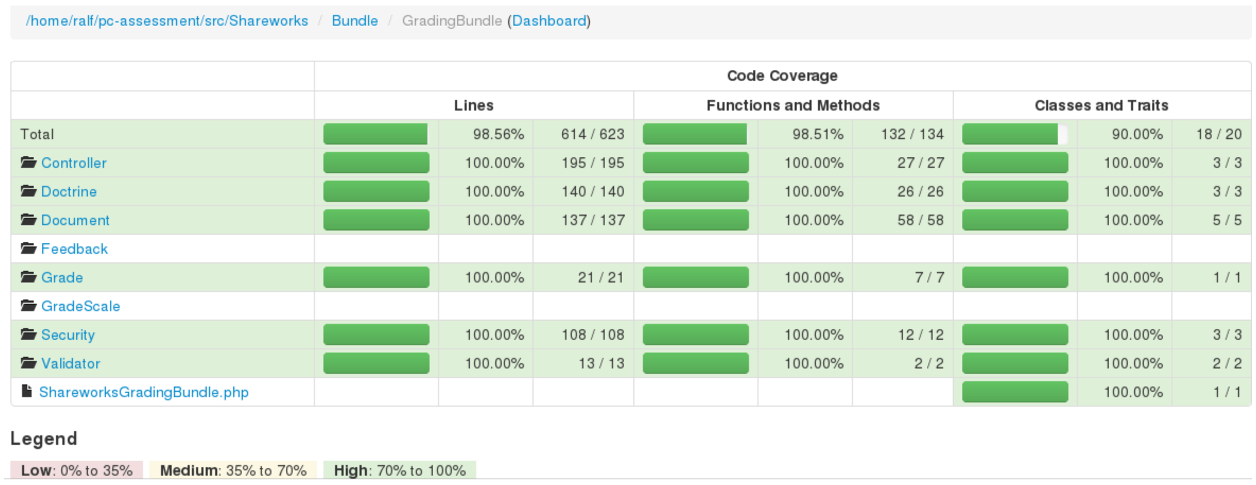
6.3 Coverage report

PHPUnit provides a framework that creates a coverage report, which tells you for each line of code by which test case(s) it is covered. Line coverage is not the best metric for asserting that your tests are good. It does tell you, however, that you do not have useless code, if you get 100% coverage without useless test cases. The best method of testing is to make sure that all code paths have been tested. This means that there is a test for each different way the code can be run. In many cases the project group made sure to also meet this constraint. For this project, a goal was to get 100% line coverage for the GradingBundle. When the code was deliverd to SIG [4] for the first time, all lines, functions, and classes were fully covered. The most important thing of testing is to assure that all paths end in the expected result. This is not assured when you have 100% coverage, but when you do not have 100% coverage, you can be sure that you do not have all border cases covered, or you have redundant code.

A fully green code coverage report, as seen in Figure 6.1, also gives you a nice spearpoint in presentations. You can say your code is fully covered by the test cases. Especially in the case of the Security folder it is required to have 100% coverage, or people would think your code is not safe.

At the moment the code is 100% covered. For some classes such as the voters this was quite difficult as functional tests would never reach some code. In these classes default cases exist that are only written because it adds clearity to the code making it more maintainable. Sometimes these default cases can never be reached by functional tests, but can only be reached by unit tests. This required some extra thought to

[1]section 3.3



/home/ralf/pc-assessment/src/Shareworks / Bundle / GradingBundle (Dashboard)

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total		98.56%	614 / 623		98.51%	132 / 134		90.00%	18 / 20
Controller		100.00%	195 / 195		100.00%	27 / 27		100.00%	3 / 3
Doctrine		100.00%	140 / 140		100.00%	26 / 26		100.00%	3 / 3
Document		100.00%	137 / 137		100.00%	58 / 58		100.00%	5 / 5
Feedback									
Grade		100.00%	21 / 21		100.00%	7 / 7		100.00%	1 / 1
GradeScale									
Security		100.00%	108 / 108		100.00%	12 / 12		100.00%	3 / 3
Validator		100.00%	13 / 13		100.00%	2 / 2		100.00%	2 / 2
ShareworksGradingBundle.php								100.00%	1 / 1

Legend

Low: 0% to 35% **Medium**: 35% to 70% **High**: 70% to 100%

Figure 6.1: Coverage Report

make sure these cases were also covered. The default cases mostly included giving exceptions as this cases should never happen.

Chapter 7

Software Improvement Group

In the bachelor's project, it is customary to have the project code checked by the Software Improvement Group, **SIG** [4]. SIG looks into all the source code that is written by the project members, and compares it to their maintainability model. The code then receives a score of at least one to a maximum of five stars, based on criteria such as volume, duplication, unit size, unit complexity, unit interfacing, module coupling, component balance and component independence. There are no checks for consistent code style, and they are unable to run the provided tests (both unit-tests and functional tests) because the code from Shareworks was not provided to them.

The project code is sent to SIG twice. Once halfway the project, after 4 scrum weeks, and once when the code was completed, after 7 scrum weeks. At this point the advice by SIG could still be used to improve the code.

The following sections explain the analysis by SIG.

7.1 Initial review

In the initial review, the grading and review module got a score of almost four stars, which means the code is above average maintainable. The highest score was not achieved because of lower scores for unit size and component balance. The presence of test code was promising to SIG, even though they could not run the tests nor create a coverage report. For the exact words of SIG (in Dutch), see Appendix C.

7.1.1 Unit size

Unit size checks on large methods, and according to SIG there were some methods that did too much, which resulted in too many paths, and therefore created untestable methods. Examples of too large methods were the *vote* methods in the FeedbackVoter and GradeVoter. These methods determine if a user is allowed to give feedback on or grade an item. Because there are a lot of possibilities, a lot of checks have to be done to come up with the right permission.

Following the feedback from SIG, the *vote* methods have been split up in smaller pieces, each piece representing a specific action (view, edit, delete), and these pieces have been tested separately, to avoid having to traverse all the created paths from beginning to end.

Several other methods have been split as well, and some duplicate code has been extracted to a separate method, or to an abstract class, to improve re-usability.

7.1.2 Component balance

According to SIG, there is no clear component structure visible in the file system, which gives the lower score on component balance. The directory structure is clear, but it is based on categorisation of code and not on logical components. Also, the naming was not really correct. 'src' and 'web' should be named 'backend' and 'frontend', because the web folder contains sources as well.

The used structure is the structure prescribed by the Symfony2 framework, that was already used by Shareworks when the project started. This structure was maintained, until the back-end and front-end were split up into two separate projects, after which the 'src' map is on the back-end repository, and the 'web' map on the front-end repository. The web folder contains the AngularJS structure of maps, which is also maintained. In the final review, the naming of directories should be better because of the split.

For the component structure, nothing will change in the final review. The structure can be quite clear, provided there is a proper explanation. Therefore, an explanation of the structure will be given here. In Symfony, all code is divided into Bundles. Each bundle can be switched on and off in the project, and each bundle should not be dependent on too many other bundles. Because the grading and review module should be a standalone module, a new GradingBundle was made in the existing project. Inside this GradingBundle, there are several directories, and therein the components are categorised. For example, all controllers are stored in the controller directory, managers (communicators with the MongoDB database via Doctrine) are stored in the doctrine folder, and documents ("items" that are stored in the database, in this case grades, feedback and grade scales) are stored in the document folder. Interfaces are stored in a folder with the name of the component they belong to, this is done to capture the model in a place where it seems most logical. The tests are in the 'tests' folder, containing the same directory structure as the normal bundle structure.

7.2 Final review

The final review did not include a score for the total system but focused more on the improvements in the system since the first review. SIG observed that the score for maintainability has a light increase and that the partial scores for component balance and unit size, being the weak points of the first review, have strongly increased. The amount of testing code has increased together with the amount of code written which gives the impression of the system being well tested. For the exact words of SIG (in Dutch), see Appendix D.

7.2.1 Unit size

About unit size SIG mentions that the amount of too big methods is a lot smaller than at the time of the first review. This is mostly achieved by splitting methods in smaller pieces, also creating more code re-usability. The partial score for unit size is now at three stars of five. This means that for unit size the score is average.

7.2.2 Component balance

According to the advice the back-end and front-end have clear names this time. SIG also observed this in the final review. The advise of SIG even has been followed literally, which was not necessary for SIG, a functional name for the two directories would be fine too. This improvement was made possible by the split that Shareworks did during the project. To read more about this split, see section 5.1

Chapter 8

Discussion

As can be read in chapter 2, Projectcampus was an already running online learning environment. What was missing was a grading and review module. The rest of this report describes the designing and implementing of this module, in a way that is efficient, maintainable and user-friendly.

In section 3.2 the requirements are discussed. Some of these requirements are the result of changes during the process of implementation.

For example there were some changes in requirement 1. The front-end is able to switch off but only by switching off the grading or feedback for the course or the assignment. This is done because disabling grading for the whole system would not be something that is useful. Switching grading of for only one organisation can be useful for Shareworks if they make it a separate paid module, this functionality is only not yet implemented as the organisation structure is not yet implemented in Projectcampus. The function of disabling grading by course or assignment comes in handy for teachers that do not need the feature in their course. This takes away expectations from students that their assignment will be graded and by that enhances user-friendliness.

During the process the group also discovered that the private notes defined in requirement 7 and 7a needed change. The private messages would need another extra stream only viewable for staff and coaches. This would end up creating too much message streams for users. Users would not know which stream to use for which action and would end using wrong message streams. Because of this problem requirement 7 and 7a have been removed to increase user-friendliness.

In the first phase of the project, creating the object model in Projectcampus, the group also discovered that requirement 14 had to be added. By only implementing grades on project level users would end up getting grades they did not work for, or lose grades they did work for. By adding this requirement the grades should work as expected, which increases user-friendliness.

After implementation of the requirements and keeping to the constraints (section 3.3) the solution became efficient, because as little as possible data is sent over from the server to the client, and this information is used to display the review information in a clear way. The solution is maintainable, because all of the code is well-documented, and code re-usability is high. In chapter 9 about recommendations, Shareworks will be given some tips to make use of this re-usability. Also the code has been kept separate from the original project code, so it will be clear what code belongs to the grading and review module. The grading and review module is also thoroughly tested, as explained in chapter 6, making bugs easy to detect. The solution is user-friendly, because both teachers and students will have all the overviews they need to give and view grades and feedback, and all this extra functionality only requires the extra steps you would expect from such a system, while also making sure the solution reacts on user actions as one would expect.

After the course of this project, the project group hopes Shareworks will gladly accept the grading and review module. Further the group hopes that upon release of the module on the live version, a newsletter will be sent to all users to make sure they know about the existence of the new functionality. The grading and review module might be used as a pay-to-use service, when Shareworks has implemented the institutional-specific environments.

The project group was not all unfamiliar with working in a real company, but they certainly got more experienced. Working within a company brings some more stakeholders at play, and gives responsibilities, for example the responsibility to represent the Delft University of Technology, and to showcase what is taught there.

The group did recognise situations where the value of certain processes taught at the university are underestimated by Shareworks. These processes are very likely to be underestimated by companies though. The most important process that is being underestimated is testing. The group does think that by their presence and demonstration of test driven development Shareworks has made a great step at it themselves. The latest newsletter of Shareworks did include the phrase that Shareworks will be using more test driven development to give the system more reliability.

As with every project, the group learned from the process of creating the product, but the group also has the idea that Shareworks learned something from them. This will generally consider organisational and process aspects but these are usually the most difficult to master.

Chapter 9

Recommendations

In the project several situations emerged where the group spotted opportunities for the future. These opportunities have been translated into recommendations for Shareworks.

9.1 Re-usable features

Some features written by the project group are re-usable in other parts of the Projectcampus system. Re-usability in a system will always increase maintainability in the system which is what was stated in the research question as something that is needed.

One of these features is the permissions service in the front-end. The recommendation for the permissions service is to use this throughout the whole front-end. It is also wise to extend this service to include all the permission checks. By doing all the permission checks on one place many code duplications can be removed, making the code more maintainable.

In the front-end the modal to open submissions in, is also implemented into a directive. Advised is to use this directive on all places where the submission modal is opened. This reduces code duplication and ensures the functionality of the submission modal that was implemented by the project group will be usable everywhere.

There is also a filter implemented to truncate pieces of text to a limited amount of characters. This filter is efficient to use in for example displaying short pieces of text with a read more button. In systems such as Projectcampus very much alike Facebook these situations might often occur. For this situation the filter that is built can be re-used.

The administration page included in the front-end can also be extended. The grade scale center is already available in the administration page, but extensions such as user, course, workspace and project management can easily be added. Such extensions on the administration page will decrease the amount of work needed to maintain the running system.

Concerning the back-end, the test framework is probably the most important re-usable feature implemented. As Shareworks did not have a test framework yet, the test framework written by the project group can be easily used. It implements several lightweight test cases and unit test cases that use a lot less overhead than normal Symfony2 test cases. It is highly recommended for Shareworks to create more tests for their code to decrease the amount of bugs found on the live environment. More tests will always mean that when programming new features the tests will warn the programmer that an existing feature has been broken. More on testing can be found in chapter 6.

Further recommendations for re-usability were to use the SubmittableItemTrait and AbstractVoter created by the group. However these classes were already implemented by Shareworks during the project. This is where insights met and which resulted in some double work. Because of the implementation of the classes by Shareworks this is no longer a recommendation.

9.2 New features

The group also has some recommendations on new features for Projectcampus. Some about the grading and review system, and some about other parts of the system.

A feature of the grading system that would be good to add is the option to disable the grading system on organisation. This can be done by adding a boolean to the organisation settings to tell if the module has been turned on. The permissions service can then check if the grading module should be displayed or not. This feature allows Shareworks to sell the grading module by enabling or disabling it on organisation level.

Once organisations are possible on Projectcampus the group also recommends to create a grade page on this level. This level should implement a browsing of all the students in the organisation, so an organisation manager can select all grades of a student. At this moment an organisation does not have an option yet to check for grades of a user. To make sure the grade system can be used without another system backing it up this option should be available. For educational institutes this gives the ability to check if a student is permitted to get his degree. This can later be extended by giving the ability to assign European Credit to courses. A report of a student can then be generated to see how many EC he or she has got and how many are still needed. The grading system will then be fully applicable to use for grade administration. For organisations this will add more user-friendliness and enhance the efficiency of the system.

A help center in Projectcampus might also be a good add-on. The help center can give more clarity about how the whole program works. As Projectcampus gets more and more features it is difficult to get to know all these features. Therefore a help center can explain the features available. The help center can then also include a section on the grading and review module. A help center in Projectcampus will create a more user-friendly environment for the user. The user then has a place to go if something is not understood.

9.3 Improvements

There is also some room for improvements in Projectcampus. These are mostly minor improvements although the missing functionality can be annoying.

Since the last refactor of the code, one week before the project ended, assignments can also be created at workspace level. The grading and review module was not accommodated for this change and might break on this case. This use case should be thoroughly tested to make sure the grading and review module does not break before the code goes live.

Another unclear case is when a user submits more than one solution to an assignment. It is not yet clear what the system should do in such a case. Within Shareworks, there is no policy on how to handle multiple submissions for the same user. The recommendation is to think of a policy for such cases and implement these into the system. The grading and review module does weigh these duplicate submissions when calculating the average, but it only displays grades for the newest submission. There are other options such as always using the submissions which scores highest. This has not been implemented yet because Shareworks' policy for double submissions is not developed yet. Not having a policy for these kind of situations results in worse maintainability of Projectcampus which is something that should be solved. By creating policies every developer knows exactly what to expect from the already existing software.

Some additions might also be good to Projectcampus at the point of generating feedback to the user. Sometimes a button is clicked but nothing happens, this can be confusing for a user. The user might go on clicking the button and thereby generate error after error. It is important to display structural feedback to the user about what is going on. The system might otherwise not be trustworthy enough for some users resulting in less customers. This improvement is advised to be placed high on the priority list of Shareworks to increase the user-friendliness of Projectcampus.

Bibliography

[1] Shareworks. `http://shareworks.nl/`. [Online; accessed November 2013].

[2] Terry Anderson. *The theory and practice of online learning.* AU Press, Athabasca University, Edmonton, 2008.

[3] Projectcamp.us. `http://projectcamp.us/`. [Online; accessed November 2013].

[4] Software Improvement Group. `http://sig.eu/`. [Online; accessed November 2013].

[5] Blackboard. `http://www.blackboard.com/`. [Online; accessed September 2013].

[6] Facebook. `http://www.facebook.com/`. [Online; accessed September 2013].

[7] Albert L. Harris and Alan Rea. Web 2.0 and virtual world technologies: A growing impact on is education. *Journal of Information Systems Education*, 20(2), 2009.

[8] Gary Chin. *Agile project management: how to succeed in the face of changing project requirements.* AMACOM Div American Mgmt Assn, 2004.

[9] Craig Larman. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, 3/e.* Pearson Education India, 2012.

[10] Scrum. `https://www.scrum.org/`. [Online; accessed November 2013].

[11] David Chelimsky, Dave Astels, Zach Dennis, Aslak Hellesy, Bryan Helmkamp, and Dan North. *The RSpec Book: Behaviour-Driven Development with RSpec, Cucumber, and Friends.* 2010.

[12] Less - the dynamic stylesheet language. `http://www.lesscss.org`. [Online; accessed November 2013].

[13] Mark Masse. *REST API design rulebook.* O'Reilly, 2011.

[14] Mongodb manual - query documents. `http://docs.mongodb.org/manual/tutorial/query-documents/`. [Online; accessed November 2013].

[15] Thomas M Connolly. *Database systems: a practical approach to design, implementation, and management.* Pearson Education, 2005.

[16] Bootstrap - modals. `http://getbootstrap.com/javascript/#modals`. [Online; accessed November 2013].

Appendix A

Project Plan

Project Plan

TI3800 Bachelor project - Shareworks (Projectcampus)

G.H. Kuijer R. Nieuwenhuizen M. de Vries

September 20, 2013

Contents

Introduction

Shareworks is an ambitious startup with a focus on educational software. Shareworks' main product is an e-learning platform called Projectcampus for project based education. The platform has recently gone live but is still missing some functionality and features. One of these missing features is a grading and reviewing system for submitted assignments. That feature will be implemented as part of this project.

This report outlines the goals and boundaries of the project, and explains how the project group will work. The first section explains the current Projectcampus system and the planned features in more detail. Section two describes the project organization and workflow, while section three contains an initial task list and time planning. The final fourth chapter explains how the project group will ensure the quality of the delivered work.

1 Project Description

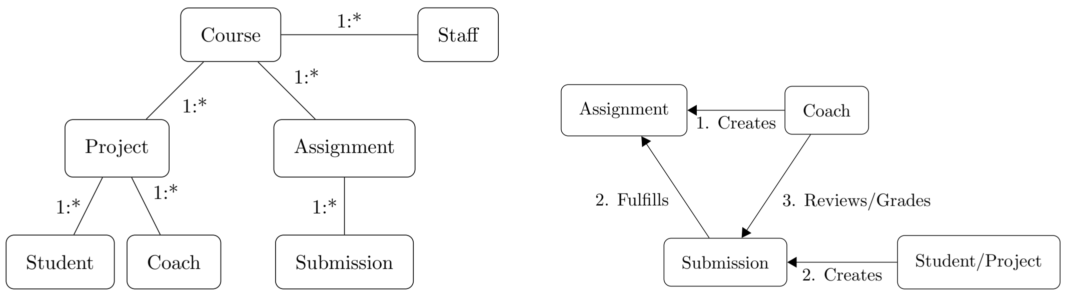
This section describes the existing system and the details of the project. It includes a list of requirements and constraints for the project.

1.1 Project environment

The existing Projectcampus system allows students to easily share information on projects they're working on. Teachers and other students can view those projects and give feedback or gain insights for their own projects. The system is similar to Facebook, but with a focus on project based education. However, it is still missing a system for teachers and coaches to review and grade the work submitted by students.

The main entities in the Projectcampus system are courses, projects and assignments. Teachers can create courses and add assignments under a course. Students can then create projects to fulfill those assignments in a group. Alternatively, assignments can be configured as individual assignments in which case students have to submit the work alone.

Each project can also have a number of coaches assigned to it. These coaches can provide feedback and reviews to help the students and to relieve the teacher of some of the work. The object model is represented in Figure 1a and the workflow is depicted in Figure 1b.



(a) Object model (b) Workflow

Figure 1: Schematics of the Projectcampus system.

1.2 Project goal

The goal of the project is to add a grading and review system to the Projectcampus platform. This is currently one of the more important features still missing from the platform. The grading and review system should allow teachers and coaches to view submitted assignments, leave feedback on them and grade them. Adding this feature will make Projectcampus a more complete solution for project based education.

1.3 Project details

The aim of the project is to create a grading and review module for the existing Projectcampus system. Since the existing Projectcampus system is separated in a back-end and a front-end, the created module must also follow this pattern. The two different components will communicate with an API that still needs to be designed. This API must be consistent with the existing API of the Projectcampus system, and all additions and changes to the API must be documented.

The front-end must be consistent with the rest of the Projectcampus system. The group can rely on Shareworks for required icons and images to maintain a consistent style.

Adding grades and reviews must be possible with minimum effort on part of the user. The module must support different grading types, such as grades from 1 to 10, A to F, pass or fail and more.

The group will employ test driven development and report progress in weekly meetings with Shareworks.

The different components of the module will be described in the next few subsections.

1.3.1 Review panel

The review panel is visible in the sidebar when a teacher, coach or project members views a project page. With this panel, the coaches can assign grades, leave feedback or leave private notes. Private notes are only visible to other coaches and teachers.

At the same time the panel provides an overview of the submitted assignments in the project for the students, coaches and teachers. This overview will show the received grades and feedback, as well as submitted assignments that still need to be reviewed and assignments that still need to be submitted. For a prototype of this panel see Figure 2.

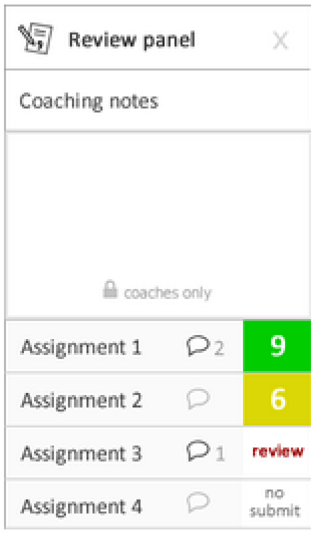


Figure 2: Review panel

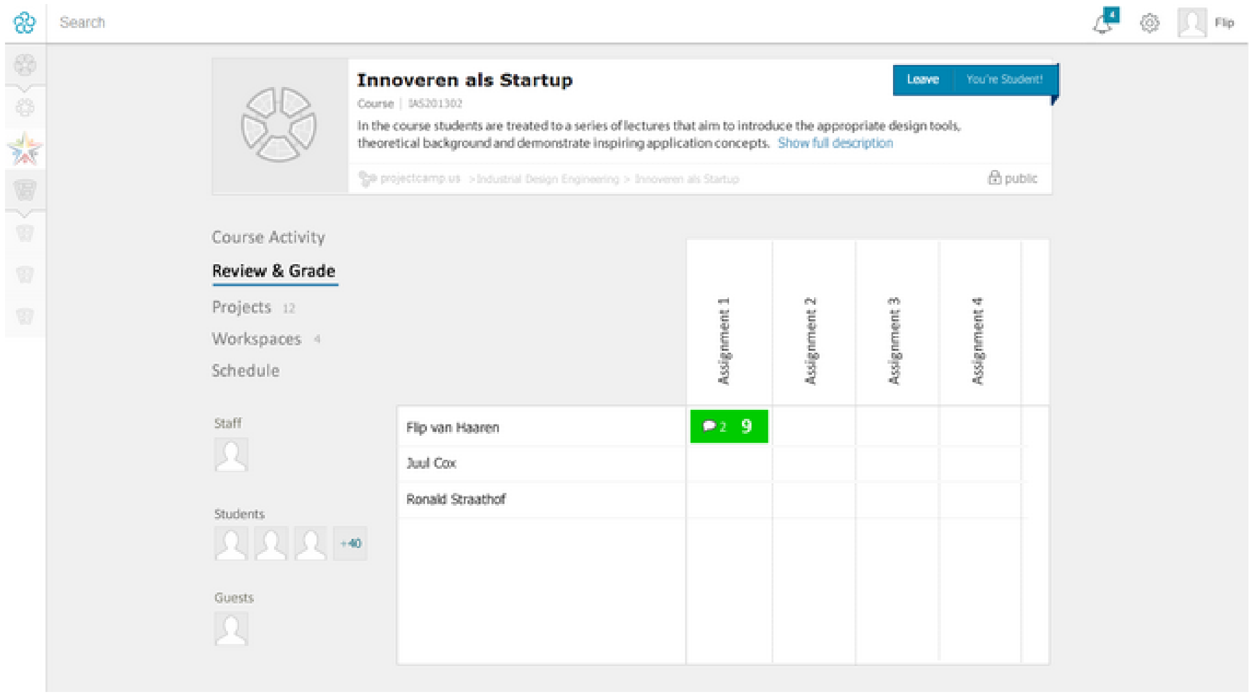
1.3.2 Grade overview

Students can also access a grade overview which shows them all their past grades, possibly including the average for each assignment so the student can compare his results. Students should be able to search trough the grade overview for specific projects and courses.

1.3.3 Assignments overview

Teachers can access an assignments page that shows them a full overview of all students, including the grades for those students for each assignment in the course. An empty spot in the table means that a student has not submitted the assignment yet. Just as with the review panel, assignment submissions that still need to be reviewed also show up.

The teacher can choose not to publish reviews immediately, but instead publish all reviews at once to prevent some students from receiving their review before others. This can be configured per assignment. As soon as a review is published, the relevant student will receive a notification. Published reviews are only visible for the students receiving the review. For a prototype of this overview see Figure 3.



Innoveren als Startup

Course | IAS201302

In the course students are treated to a series of lectures that aim to introduce the appropriate design tools, theoretical background and demonstrate inspiring application concepts. Show full description

projectcamp.us >Industrial Design Engineering > Innoveren als Startup

	Assignment 1	Assignment 2	Assignment 3	Assignment 4
Flip van Haaren	2 9			
Juul Cox				
Ronald Straathof				

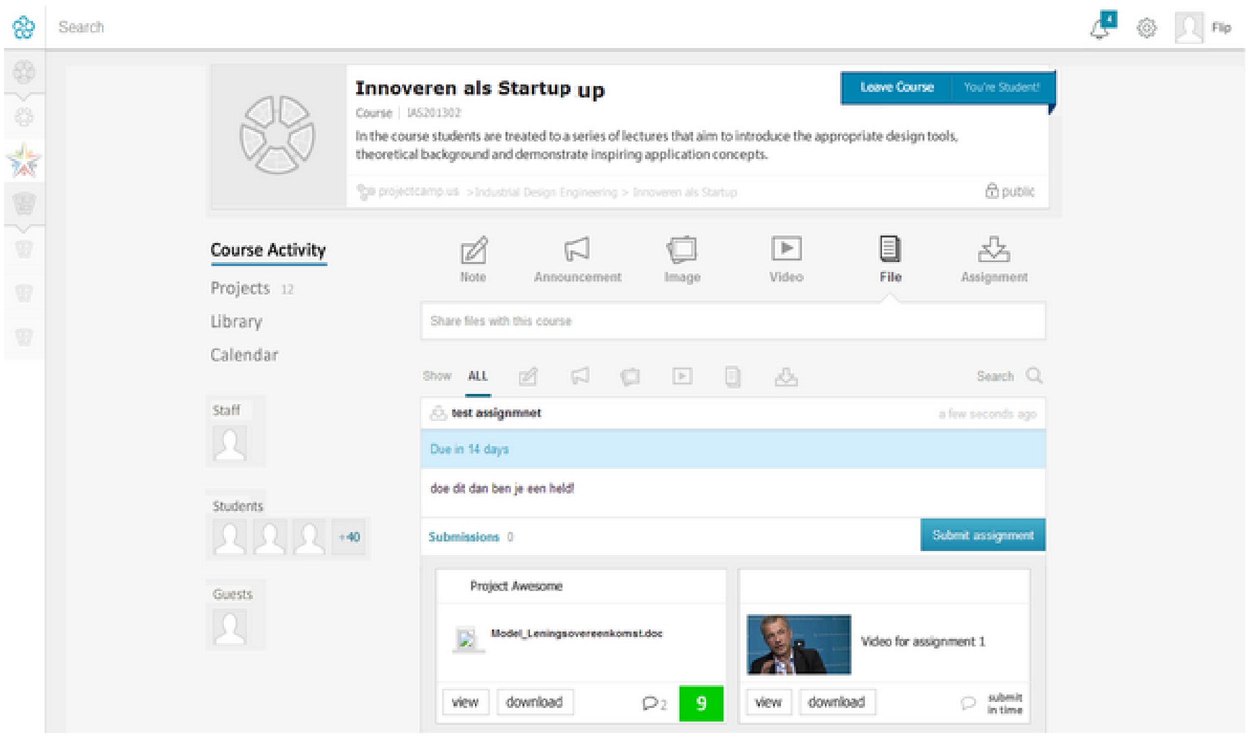
Figure 3: The review & grade section

1.3.4 Course activity

The course activity feed will also display grades and reviews students. Assignments are already listed in the activity feed. This existing view should be extended to show students their submissions with the assignment, including grades and reviews. The reviews can be read in a tooltip and teachers can add comments using a popup. Students will only see their own submissions, grades and reviews in this manner. For a prototype of this feed see Figure 4.

1.4 Deliverables

The finished product is a module for the Projectcampus system that enables grading and reviewing of projects and assignments. The module will be delivered with tests and documentation. The project group will also write an orientation report and a final report for the TU Delft.



Innoveren als Startup up

Course | IAS201302

In the course students are treated to a series of lectures that aim to introduce the appropriate design tools, theoretical background and demonstrate inspiring application concepts.

projectcamp.us > Industrial Design Engineering > Innoveren als Startup

public

Share files with this course

test assignmnet

a few seconds ago

Due in 14 days

doe dit dan ben je een held!

Submissions 0

Submit assignment

Project Awesome

Model_Leningsovereenkomst.doc

Video for assignment 1

Figure 4: The course activity feed

1.5 Requirements

The desired product must meet some requirements, established in consultation with the client. Below you will find a list of these requirements.

- The grading and reviewing system must be a separate module that can be switched on and off.
- The grading and review interface must be available through a review panel in the sidebar on the project page.
  - Project-based assignments must appear in this panel.
- Teachers and coaches must be able to grade submitted work.
  - Both user-based and project-based assignments can be graded.
  - Different kinds of gradings can be given (dynamic, 1-10, pass/fail, A-F, +/-)
  - It must be possible to edit grades.
  - Grades will only be visible to students after publishing by the coach or teacher.
  - It must be configurable to publish grades delayed or instantly.
- Teachers and coaches must be able to review submitted work.
  - Both user-based and project-based assignments can be reviewed.
  - It must be possible to edit reviews.
  - Reviews will only be visible to students after publishing by the coach or teacher.

    - It must be configurable to publish reviews delayed or instantly.
- It must be possible to publish grades and reviews separately.
- Students must be able to read reviews.
- Teachers and coaches must be able to share private notes with each other.
    - The notes must be stored on autosave.
- Students must be able to view all their grades in one overview.
    - Students must be able to search for specific courses and projects through their grades.
- Teachers must be able to view all assignment submissions in one overview.
    - The overview must include grades and reviews when available.
    - The teacher must be able to view either user based assignments, project based assignments and both mixed in one view.
    - Project based assignments must be visibly distinct from user based assignments when mixed.
    - Project based results will count for all students in the project in the mixed view.
    - After clicking an assignment, a dynamic review panel for this user-based assignment must open.
        - Switching between users with a forward/back button should be possible.
- Published grades should show up in the course activity feed.
- All actions should provide visual feedback to inform the user of progress or failure.
- Grades and reviews should generate notifications for students after publishing.
- It should be possible to check submitted assignments for fraud.

1.6 Constraints

During the process, both the group members and the client must honour some constraints to keep the project going smoothly. The list of these constraints is given below.

- The front-end and back-end must be separate systems.
- The front-end and back-end must communicate using an API.
    - The API should be consistent with the existing Projectcampus API.
    - The API should be well documented. For each API call, the documentation should include:
        - A description of what the API call does, including pre- and postconditions.
        - A description of every parameter passed to the API call, including it's type and what it is used for.
        - A description of the returned value of the API call.
- The front-end must provide a look and feel consistent with the rest of the Projectcampus front-end.
- The front-end and back-end must be thoroughly tested.
    - The back-end must have unit tests in Symfony.
    - The front-end must be dynamically tested with AngularJS.

- The system must be secure against unauthorized access and modification.

2 Process and Planning

This section explains the development process that will be used and give an initial planning for the project. It will start by explaining the development method for the project and continue to give an initial assessment of the major tasks in the project. Finally, it will provide a rough outline of the project.

2.1 Methods

The group will use the Scrum software development method for the project. Scrum is an agile programming method where the team works in iterations (called sprints) of one or two weeks. After each sprint, the team delivers a working product that has improved since the previous sprint. The project group will work with sprints of one week.

All sprints will start with a planning meeting where the team will decide what tasks will be worked on during the sprint. After each sprint, the team will hold a review meeting to see what was done and what can be improved for subsequent sprints.

Each day will start with a daily Scrum meeting where the group members inform the team of what they have done since the last daily meeting and what they will do the rest of the day. If a team member is stuck on something he will inform the rest of the team, so that the team can decide how to proceed best.

The team will also employ test driven development. This forces the team members to think how to implement a feature first, write a test for the feature and finally implement the feature. This provides an easy way to confirm that features are working as they should and prevents tests being added as an afterthought.

2.2 Tasks

See below for a list of the major tasks that have to be completed for this project.

- Back-end:
  - Design object model for grading and reviewing system.
  - Extend existing object model for new features.
  - Design and implement API for making, editing and deleting reviews.
  - Design and implement API for making, editing and deleting grades.
  - Design and implement API to search and retrieve grades and reviews.
  - Extend existing API to support (delayed) publishing of grades.
  - Create email feedback to users from which grades have been published.
- Front-end
  - Design and implement grade and reviewing panel.
  - Alter existing assignment view in course feed to show grade and review information.
  - Design and implement grade and review overview panel for students.
  - Design and implement grade and review overview panel for teachers and coaches.

2.3 Planning

The initial planning for the project can be found in the table below.

Weeknumber	Global task	Partial tasks
38	Orientation	Compile project proposal
		Create project plan
		Compile project requirements
39	Orientation	Create orientation report
40	Work	First iteration
41	Work	Second iteration
42	Work	Third iteration
43	Holiday	
44	Work	Fourth iteration
		Send code to SIG
45	Work	Fifth iteration
46	Work	Sixth iteration
		Create final report
47	Work	Complete and send final report
		Send code to SIG
48	Work	Endpresenation

3 Project Organization

This section describes the organization of the project. It explains the distribution of roles within the project group, the qualifications of the project members as well as the financing of the project, how the group will communicate progress and problems to the client, and what technologies and resources the group will use during the project.

3.1 Distribution of roles

In principal, all project members perform the same tasks within the project, including API design, programming, documenting and testing. Every team member is responsible for the quality of the delivered work and reports. However, Gijs Kuijer will also take on the role of Scrum master and Ralf Nieuwenhuizen will be the head editor of the written reports.

3.2 Group

All group members will invest 40 hours per week on the project. The team will work on site at Shareworks at their office in Rotterdam. They will also familiarize themselves with the programming languages, frameworks and techniques used during the first two weeks of the project. The group members will document their work where required, and write tests for all parts of the system they create.

3.3 Finance

Group members are not expected to have any expenses as part of the project. Should the need arise to spend money on the project, the costs will be checked by the group first and then forwarded to the client.

3.4 Reporting

Progress will be reported to the client in weekly meetings. Additionally, the team will be working on site at Shareworks for most of the time, so any immediate problems will be discussed as they arise. The client will be given a chance to review and comment on all reports before they are handed in to the university. All reports will be presented to Shareworks before handing them in with the university.

3.5 Technologies

The project will be implemented in different programming languages and frameworks for the front-end and back-end. The back-end will be written in PHP using the Symphony2 framwork and uses a MongoDB database. The front-end will consist of HTML with Javascript and CSS and it will use the AngularJS framework. Instead of regular CSS, the project will use LESS[1]. All of these technologies are already decided on by the client, as the system is already running live.

Reports for the university will be written usin LaTeX, while documentation for the client will be maintained on Google Drive.

[1]`http://lesscss.org/`

3.6 Resources

The team members will work on their own laptops, generally at the Shareworks office in Rotterdam. Shareworks may provide additional monitors to connect to the laptops. The client will also provide the team with a license for PhpStorm (an IDE for PHP) and a central git repository on Bitbucket. Other resources include PivotalTracker for progress monitoring and Google Drive for sharing documents.

4 Quality Assurance

To ensure that the quality of the project will meet the set standards, measurements must be taken. These measurements will be explained in the following sections, divided into documentation rules, version control, and evaluation.

4.1 Documentation

A good project needs to have a proper documentation. Documentation must be written for the data model, the API between the front-end and back-end, the back-end itself and the front-end itself. In the very first stage of the project this project plan was written to describe the project, work flow and time management. When the research and orientation is done an orientation report will be written which will be containing amongst others the system design.

4.2 Version control

Code will be checked in to a central git repository on Bitbucket. This allows the team to retrieve old versions of a file should it be necessary. Commits will be kept small with a clear commit message. This will help keep the version history meaningful and informative, and it will make it easy to revert single changes.

4.3 Evaluation

As part of the Scrum software development method, every sprint will end with an evaluation. This will help detect problems early so they can be solved for subsequent sprints. Shareworks will also have the opportunity to discuss progress and provide input in weekly meetings. Similarly, the group will hold regular meetings with the TU Delft advisor to evaluate progress. At the end of the project the group will write a final report documenting the entire project, including an evaluation of the product and process.

Appendix B

Orientation Report

Orientation

TI3800 Bachelor project - Shareworks (Projectcampus)

G.H. Kuijer R. Nieuwenhuizen M. de Vries

September 27, 2013

Contents

Introduction

This report documents the first two weeks of orientation of the bachelor project at Shareworks. The first two weeks were reserved to create a good view of how the current software product is built up and how it works. Furthermore, the research question had to be answered before the implementation phase could start. The research question can be found in the following quote:

> What is the best solution to design and implement an online grading and review system within the Projectcampus system that is both efficient, maintainable and user friendly?

The goal of this project is to create a grading and review system for Projectcampus, an online education environment for project based learning that is being built by Shareworks. Shareworks wants to deploy Projectcampus through the whole of Europe and later throughout the world. This means that every part of the system should be made with international compatibility in mind. Most notably, different grading scales used throughout the world should be supported. For more information on the existing system, see Appendix A.

This report will start by investigating products similar to Projectcampus. It will also explain how the system will cope with multiple grading scales, which feedback methods will be supported, what the current object model is, how it will be extended and how the front-end and API will be designed. The software and frameworks that are used in the system will be explained in the final part of the report. The report ends with a conclusion drawn from the rest of the report.

Chapter 1

Similar products

As always in the commercial world, other companies have already created similar systems, in this case an online social learning platform. Because not all social learning platforms have the same functionalities, some of them are described in the following sections. These systems are Blackboard[1], Dokeos[2], Moodle[3], Sakai[4], and itslearning[5]. Because Projectcampus also involves a lot of social interactions, Facebook[6] will be seen as a similar product as well.

1.1 Blackboard

Blackboard is a complete learning environment in use amongst others at the TU Delft. Blackboard contains different platforms, including Blackboard Learn. While certainly not the fastest, Blackboard offers a lot of functionality, including online assessment, course evaluations, a rich content editor, a user-centric interface and active collaboration. In Blackboard, courses can be created and managed by staff, and students can choose to enroll themselves. When enrolled, students are able to view the content teachers have shared with them. Blackboard also has the possible for students to communicate with each other using forums, but this isn't stimulating. As effect, Blackboard functions mostly as an information centre for students.

1.2 Dokeos

Dokeos is an open source online learning suite. It provides admin powers for management of users, courses and groups, gives students the possibility to organise video conferences and to create mind maps. It also gives teachers the tools to create online scenarios that students can follow, create quizzes and record podcasts to publish online. It seems to be a flexible product.

1.3 Moodle

Moodle is another open source learning management platform. It provides four different privilege levels, being administrators, managers, teachers and students. Courses can be created and students can take online surveys. Moodle also allows teachers to share information. Moodle does not provide tools to interact with fellow students, which makes it different from Projectcampus.

1.4 Sakai

Sakai is supported by the Sakai Foundation, in which people from MIT, Stanford, Cambridge and UvA in Holland are collaborating. The Sakai Project is open source, and can be used both on your own server, but also as SaaS (Software as a Service), like Projectcampus. Features in the Sakai Project Collaboration include announcements, a calendar, a chat with project members, an email archive and the possibility to create a glossary. So far, this is the most resembling to Projectcampus, but at the moment more complete. Some of these features could be implemented into Projectcampus in the future.

1.5 itslearning

itslearning seems like a highly customizable, complete, SaaS online learning environment. It comes with all the basic tools, and includes the Grade Book, which is similar to the goal of this project, and a plagiarism tool, one of the 'could haves' of this project. itslearning could be a big competitor for Shareworks, although it is less project-oriented.

1.6 Facebook

Projectcampus is based on project education which means that team collaboration is an import aspect. Because of this, Facebook, one of the main social media sites of this moment, can also be considered a similar product. Based on Facebook, you can comment on other people's assignments and even 'like' them. Sharing content is also an important feature of Facebook, and many different file types are supported. Notifications are also available in Facebook, a feature that is being worked on in Projectcampus. In Facebook, you can easily create a group with people sharing the same interests. In Shareworks, this can only be done by creating a project group. A nice addition would be to be able to subscribe to a certain keyword and be added to a group of people who like that too. The events feature from Facebook would also make a good addition to Projectcampus. It could be used to organise presentations or demos for a project.

Chapter 2

Grading scales

One important part of the grading and review system are different grading scales. There are different ways to represent grades. This section investigates the different grading scales in use and explains how the system will deal with them.

2.1 Problem

There are many different grading scales around the world and multiple scales can even coexist within a country. This could happen when a university focussed on international education adopts foreign standard, or sometimes adopt a different scale for no specific reason. Multiple grading scales can even be used within single educational institution. This might happen because a teacher uses his own grading scale for his course before translating it to the institution's standard, or because an assignment is a simple pass/fail assignment.

The most common grading scales use either letters, percentages, a number or a simple pass/fail. However, the exact details can vary wildly. Some grading scales allow an A- and some allow an A/B. Some add an O for outstanding and some skip the E[7]. Numbers are often in the range of 1 to 10, but sometimes the scale goes up to 20[8]. In Germany grades go from 1 to 6 where the best grade is a 1 and a 6 is the lowest grade[9]. Even the scales using percentages can have a different minimum grade to pass.

Whatever the cause of the different grading scales, a good grading system should be able to deal with them. Because of the sheer amount of variation possible in the grading scales it is unfeasible to support all grading scales in existence. The only solution that remains is to allow users to add new grading scales.

At the same time the system should be easy to use for the end users as these are generally not system experts. Luckily, adding grading scales will not be a daily task. It seems reasonable to expect at most up to 5 different grading scales in use at a single institution. The grading scales can even be added by a Shareworks employee as part of product support.

If the end users don't have to use the grading scale system directly it can be somewhat more complex too allow for more flexibility. Of course, the system should still be usable for a teacher that wants to create his own grading scale.

2.2 Solution

Considering the above, grades will be converted internally to a floating point value between 0 and 100 inclusive. A value of 0 represents a low score while a value of 100 represents a high score. The exact mapping of grades to this internal score is determined by the grading scale.

Shareworks and users can define new grading scales, although the system will contain a few default grading scales for some countries. A grading scale can be defined as either linear or symbolic.

A numeric grading scale is defined by a minimum, a maximum, a step size and a flag to indicate if the scale is inverted like the German scale. The minimum grade will be mapped to an internal score of 0 and the maximum grade to 100. Everything in between is distributed linearly.

For symbolic grading scales, every symbol has to be defined separately. A symbol is defined by the text representation and the internal score. When grades have to be averaged, the internal score is averaged and eventually mapped back to the closest matching symbol. There is no automatic support for additional plus or minus signs after a symbol because not all grading scales allow it and those that do can have different meanings for them.

Because all grades are mapped to internal scores first, it is even possible to average grades from different scales. However, this might be confusing for students and teachers, so teachers should avoid doing this.

Chapter 3

Feedback methods

Teachers should be able to provide feedback on assignment submissions. There are multiple ways to implement such a feature, so this section will investigate which fits best in the Projectcampus system. There are two general ways of giving feedback on assignment submissions: adding comments in the submitted document, or allowing feedback outside of the document.

3.1 In-document feedback

Projectcampus already uses Crocodoc to view documents (see section 7.6). Crocodoc supports adding noted to existing documents. This would make it relatively easy to implement comments in documents.

However, precautions must be taken to ensure that the notes are not visible for everyone. They should only be visible to the students in the project, the coaches and the teacher. The old document without annotations should also remain available. It is also not a requirement of the project, so it will not be on high priority.

3.2 External feedback

Another feedback method is to allow teachers and coaches to write an arbitrary text with an assignment submission. An advantage of this method is that the students can easily read all feedback without having to scan through the entire document. Of course, it does mean that the feedback has less context and the feedback might be harder to interpret.

When implementing feedback outside of the document, students, coaches and teachers should be able to reply to the feedback. This way students can ask for clarification or possibly even object. This method of feedback will be implemented in the system first. If there is time left, in-document feedback may also be added.

Chapter 4

Object model

The current data model has a number of different objects. The most important ones are courses, projects, users and items. Items can be subdivided again in announcements, assignments, blog posts, files, images, notes and videos. A simplified class diagram of the most important classes and their relevant properties can be found in Figure 4.1.

Since the system uses a document store rather than a relational database, the model is rather free. This can be seen in the owner property of a BaseItem, which can be either an assignment, course, project or workspace. While the object model might show duplicate data – such as a list of owned projects for a user and a list of owners for a project – the data is only stored with one of the two (the projects in this case) and the other is looked up dynamically when needed. Additionally, all objects have an ID that is automatically generated by the database. These have not been included in the figures.

One important part of the model is the SubmittableItemInterface. This interface must be implemented by all items that can be submitted for assignments. At the moment it only contains a get and set method for the linked assignment.

4.1 Assignment submissions

To support grading and reviews, the model needs to be extended. A simple solution would be to add a getter and setter for grades to the SubmittableItemInterface and implement them on every submittable item. However, if we do that, the only way to link a grade to a user is to check what project the submission was for and then check the owners of that project. Should the project owners change later, the grades will suddenly be linked to the wrong users.

Instead, a new grade object is added that links a grade to a submission and a user. One grade object will be created for each project owner. This way, if the project owners change, the grades will still count for the correct users. It remains possible to look up the grade for a submission and it becomes easier to look up grades for a given user.

Another alternative would be to simply store a list of users with the grade instead of one grade object per user. However, if the system should support individual grading at some later point, the model would need to change. Instead, the model is chosen to support it already, even though it is not a requirement at this moment.

These changes to the model can be found in Figure 4.2.

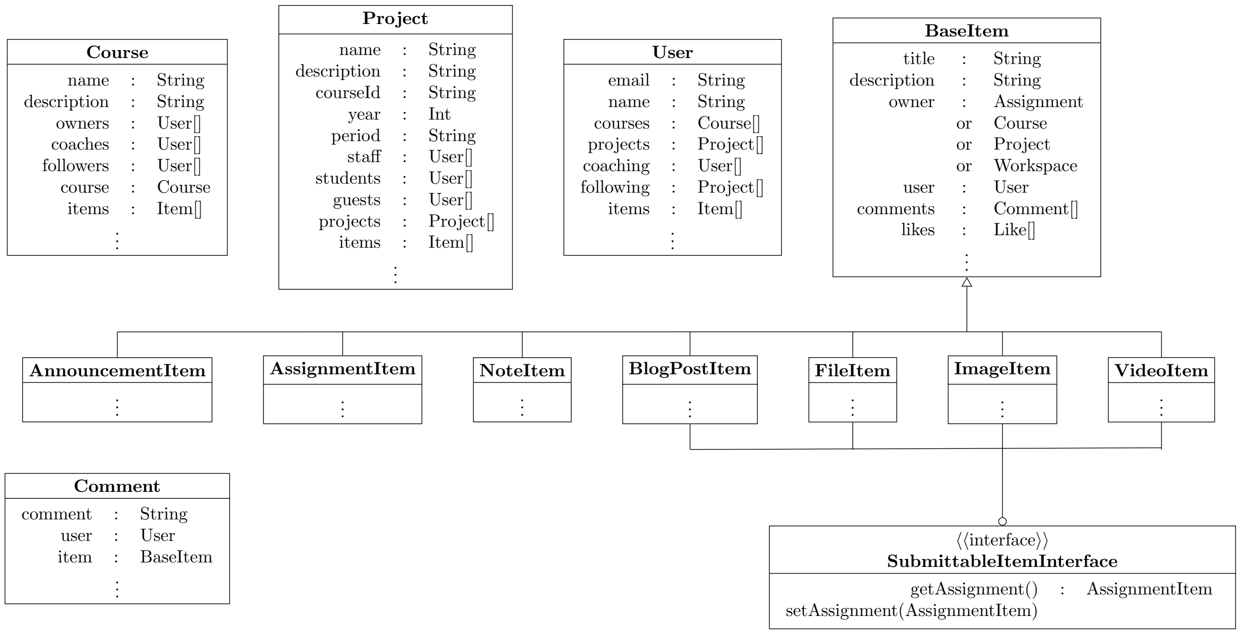


Figure 4.1: Simplified object model. Not all classes and properties are shown.

4.2 Feedback

Feedback could be stored as a property of assignment submissions. With that solution, the SubmittableItemInterface would be extended with a setter and a getter for the feedback. However, that wouldn't show who wrote the feedback unless extra properties were added. These properties are not really related to a submission directly, so they should not be added there.

Additionally, it would be impossible for students to reply to feedback if it was implemented this way. This wouldn't match the design in chapter 3. Instead, feedback will be implemented as a separate comment stream on the submission. To achieve this, a new model class is added that inherits from the existing Comment class and adds no new properties or methods. The fact that it has a different type is enough to distinguish it from regular comments. See Figure 4.2 for a diagram.

4.3 Grading scales

Because of the issues described in chapter 2, grading scales have to be stored in the database as well. See that section for more information why grading scales are represented the way they are.

There are two subtypes: linear grading scales and symbolic grading scales. Linear grading scales are defined by a worst grade, a best grade and a step. Symbolic grading scales are defined by a dictionary that maps symbols to scores. Both subtypes inherit from the same abstract base, which includes abstract methods to translate between grades and internal score but also the float value at which a grade is sufficient. See Figure 4.2 for a diagram of these classes.

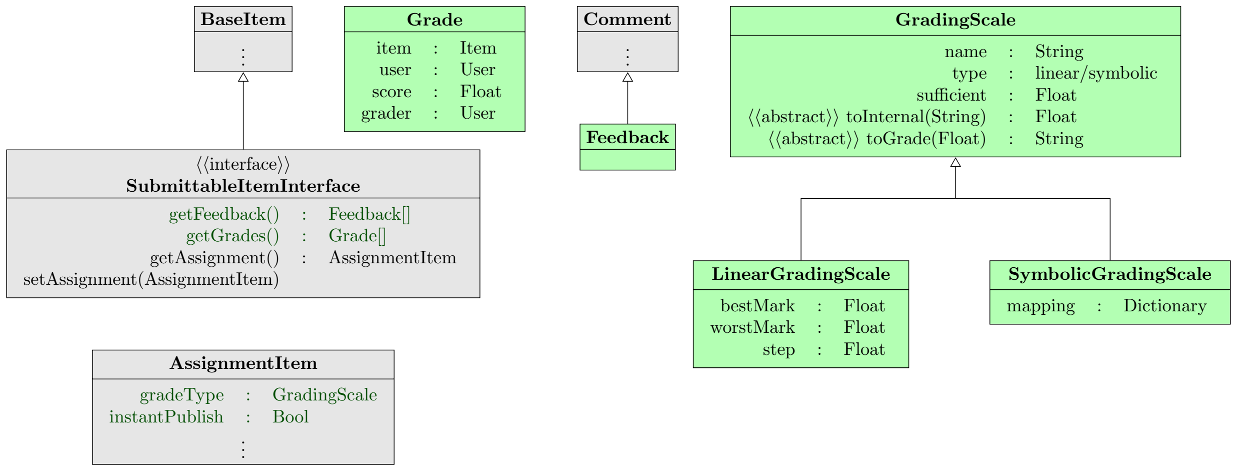


Figure 4.2: Changes to the object model. New classes and properties are shown in green.

Chapter 5

Front-end design

To make all features available to the users, new interface elements must be added to the existing user interface. The new elements must be good looking, user friendly and consistent with the existing interface.

Shareworks has already made some preliminary sketches of the interface which will guide the design. Where conventions exists in the current system they will be followed in the new interface elements. Shareworks will have the opportunity to adjust the style and interface design to ensure it stays consistent.

Students will have a grade center where they can view their grades, separated on user- and project-based grades, or mixed if they want to. The overview will include a search bar to quickly find the grades they are looking for. Grades will also be visible on the project page and course page, so students can quickly see their grades even without the grade center. Reviews will be visible in tooltips when the mouse is held over a grade.

A similar interface will be made for teachers, except that they can see all grades for all students in their course, rather than their own grades.

Additionally, a grade and review panel will be added to project pages. This allows coaches and teachers to view the project while they're writing feedback.

Chapter 6

API Design

The existing web API of Projectcampus also has to be updated for the new features added during this project. The API has to be extended to allow adding, retrieving, updating and deleting grades and reviews, as well as defining new grading scales, retrieving, updating and deleting them. The API must ensure that all constraints on the input are enforced, and that users can not perform actions or access data they are not authorized for. The API should also support publishing feedback and grades after they have been submitted, and expose all new course and assignment settings for the grading and review system.

All changes and additions will follow the same naming schemes and other conventions already used in the Projectcampus system. The changes will also be documented so that Shareworks can continue development of easily, even after the project has ended.

The functions that will be added to the API are documented in the list below. The given URL in the header is the API call, and the methods given are HTTP methods. POST implies an addition, GET implies a request, PUT implies an edit and DELETE implies a removal. After each method follows a description of what the call does for this method.

/grades/

POST	Add a new grade

/grades/{gradeId}

GET	Get the grade with id gradeId
PUT	Edit the grade with id gradeId
DELETE	Delete the grade with id gradeId

/courses/{slug}/grades/

GET	Get all grades from course{slug}

/projects/{slug}/grades/

GET	Get all grades from project{slug}

/users/{slug}/grades/

GET	Get all grades from user{slug}

/grade_scale/

GET	Get a list of all grade scales
POST	Create a new grade scale

/grade_scale/{scaleId}

GET	Get the grade scale with id scaleId
PUT	Edit the grade scale with id scaleId
DELETE	Delete the grade scale with id scaleId

/items/{itemId}/feedback/

GET	Get all the feedback from item with id itemId
POST	Add new feedback to item with id itemId

/items/{itemId}/feedback/{feedbackId}

GET	Get the feedback on item{itemId} with id feedbackId
PUT	Edit the feedback on item{itemId} with id feedbackId
DELETE	Delete the feedback on item{itemId} with id feedbackId

Chapter 7

Software and Frameworks

This section will be used to describe all the software and frameworks used to build the Projectcampus system. Of all these the description will be given what part they play in the application, how they work and why they are used.

7.1 Symfony2

Symfony2[10] is a PHP framework. It is used to program the back-end of the Projectcampus application and creates an environment in which a lot of functionalities are already present. One of the many features of Symfony is a debug environment that can monitor almost every aspect of the application.

Symfony organises the back end in bundles. Each bundle has a specific task. In general, all the other frameworks or libraries used in the back-end have a matching bundle to integrate them with the rest of the system.

Symphony handles web requests by forwarding them to the correct controller. The controller handles the request and returns a response object which will be send back to the browser. Symphony also allows bundles to provide services which can be used easily from other bundles and eventually in controllers.

Sympfony is used for the Projectcampus platform because of its simplicity and ease of learning. Another advantage of Symfony is that it is open source and thus free to use, while also having regular updates. These regular updates make sure that Symfony evolves over time and gets more functionality or better performance.

7.2 MongoDB

The database system used for Projectcampus is MongoDB[11]. MongoDB is not a relational database but a document store. As such it doesn't use SQL but has its own query language. Data is returned in JSON formatted documents which are easily accessible for the application.

The great advantage of MongoDB is its high scalability. Databases running on MongoDB can easily be mirrored and shared between networks. The querying language of MongoDB is at least as powerful as SQL, but allows more freedom in regards to what is stored.

MongoDB is used because of its few limitations and just as with Symfony it is open source and has regular updates.

7.3 AngularJS

AngularJS[12] specifies the front-end of the Projectcampus application. It is a framework which makes use of JavaScript and HTML. The framework provides an easy way to load and save content while staying on the current page, which reduces load time and can enhance the usability of the website. AngularJS also extends the already rich features of JavaScript with some more functionality such as automatic form validation.

AngularJS is used by Projectcampus to reduce the page loads and provide a smooth user experience. It provides easy front-end development and it is open source, just as the previous frameworks.

7.4 Doctrine

Projectcampus uses Doctrine as a persistence layer. To be specific, it uses the Doctrine MongoDB project[13], which is an object document mapper (ODM). It handles the database queries and provides an easy interface to store and retrieve objects from the database. Again, Doctrine is open source software.

7.5 Elasticsearch

Elasticsearch[14] is the search engine used in Projectcampus. It maintains an index of specified fields of data objects and provides fast searching through the indexed fields. Elasticsearch does not return the entire object in response to a search query, but it will give the ID's of the matching objects. Those objects can then be fetched from the database regularly.

Elasticsearch is used in Projectcampus to allow fast searching through content. This software is also open source.

7.6 Crocodoc

Crocodoc can convert many different file types for display in an HTML5 browser. This gives the users the ability to view all kinds of content without downloading them manually and even if they don't have the software to view the content.

Crocodoc is operated by passing a document to the server of Crocodoc, which will then convert it and pass it back. This document is then cached for later use.

The software is used to make documents easily available to users and to solve access problems by users. Crocodoc is not open source but asks a fee per converted document.

7.7 How it works together

The best point to start is the user interface. The user interface is created with AngularJS. This is an AJAX framework, meaning that it makes use of XmlHttpRequests to call PHP scripts and retrieve data without reloading the page. When AngularJS needs data it makes a web request which is handled by PHP, and specifically Symfony2. AngularJS makes a POST or GET request for saving or retrieving data, but the user interface normally does no hard computations.

Symfony delegates incoming requests to the correct controller, which in turn uses Doctrine to retrieve or store data with MongoDB. Similarly, controllers can invoke Crocodoc to convert documents for display in browser and Elasticsearch to execute search queries. To communicate easily with the AngularJS front end, the controllers respond with JSON objects instead of old fashioned HTML pages. Finally, the front end code using AngularJS updates the page in the browser.

Chapter 8

Conclusion

As seen above, implementing the grading and review system into Projectcampus brings some complexities and choices. In the previous sections, these problems are stated and the choices that are made were explained. To come back at the research question, the solution presented above is an interpretation of the "best solution". To get to the best result in Projectcampus, the style will be consistent with the current, which improves user friendliness. For the grading, the user will be free whether or not to use it, and they have the ability to create their own grading scales. This gives a high level of customizability, but a few default scales will be included that match common use.

Reviewing will be kept simple as a second comment stream, which gives the proper security and is easy to use. To keep the project maintainable, the API will be updated and properly documented. This way, the client will be able to adapt the product to their needs after the project ended. All modifications will use the same frameworks and software already in use, which keeps the system consistent and prevents any new dependencies from being added to the system.

Bibliography

[1] Blackboard. `http://www.blackboard.com/`. [Online; accessed September 2013].

[2] Dokeos— open source e-learning. `http://www.dokeos.com/`. [Online; accessed September 2013].

[3] Moodle.org: open-source community-based tools for learning. `http://www.moodle.org/`. [Online; accessed September 2013].

[4] Abot sakai — sakai project. `http://www.sakaiproject.org/about-sakai`. [Online; accessed September 2013].

[5] itslearning. `http://www.itslearning.eu/`. [Online; accessed September 2013].

[6] Facebook. `http://www.facebook.com/`. [Online; accessed September 2013].

[7] U.s. grading system. `http://ursinus.abroadoffice.net/Grading.html`. [Online; accessed September 2013].

[8] The french grading system. `http://www.studyineurope.eu/study-in-france/grades`. [Online; accessed September 2013].

[9] German grading system. `http://www.studying-in-germany.org/german-grading-system/`. [Online; accessed September 2013].

[10] Symfony2. `http://http://symfony.com/`. [Online; accessed September 2013].

[11] Mongodb. `http://www.mongodb.org/`. [Online; accessed September 2013].

[12] Angularjs. `http://angularjs.org/`. [Online; accessed September 2013].

[13] Doctrine mongodb. `http://docs.doctrine-project.org/projects/doctrine-mongodb-odm/en/latest/`. [Online; accessed September 2013].

[14] Elasticsearch. `http://www.elasticsearch.org/overview/`. [Online; accessed September 2013].

Appendix C

SIG: Initial review

(

Analyse)

De code van het systeem scoort bijna vier sterren op ons onderhoudbaarheidsmodel, wat betekent dat de code bovengemiddeld onderhoudbaar is. De hoogste score is niet behaald door lagere scores voor Unit Size en Component Balance.

Wat opvalt bij het bekijken van de code is dat er geen duidelijke componenten-structuur zichtbaar is op het file-systeem, wat resulteert in de lagere deelscore voor Component Balance. Er is wel een duidelijke directorystructuur, maar die is gebaseerd op een categorisering van code en niet zozeer op logische componenten. De naamgeving klopt ook niet helemaal: “src” en “web” zijn eigenlijk “backend” en “frontend”, want de “web” directory bevat net zo goed sources.

Voor Unit Size wordt er gekeken naar het percentage code dat bovengemiddeld lang is. Het opsplitsen van dit soort methodes in kleinere stukken zorgt ervoor dat elk onderdeel makkelijker te begrijpen, te testen en daardoor eenvoudiger te onderhouden wordt. Voorbeelden van (te) grote methodes in jullie code zijn FeedbackVoter.vote() en GradeVoter.vote(). Deze methodes doen eigenlijk te veel, waardoor het minder makkelijk wordt om er unit tests voor te schrijven. In het geval van FeedbackVoter.vote() zie je dat ook duidelijk in de code, omdat er binnen de switch cases ook weer ifs worden gebruikt krijg je een groot aantal mogelijke paden. Wat jullie (bijvoorbeeld) zouden kunnen doen is de inhoud van switch cases opsplitsen naar aparte methodes (voor view/edit/delete), die kun je vervolgens apart testen.

Over het algemeen scoort de code bovengemiddeld, hopelijk lukt het om dit niveau te behouden tijdens de rest van de ontwikkelfase.

De aanwezigheid van test-code is in ieder geval veelbelovend, hopelijk zal het volume van de test code ook groeien op het moment dat er nieuwe functionaliteit toegevoegd wordt.

Appendix D

SIG: Final review

Hermeting

In de tweede upload zien we dat de score voor onderhoudbaarheid licht is gestegen. De deelscores voor Component Balance en Unit Size, die tijdens de vorige analyse werden genoemd als mogelijk verbeterpunt, zijn sterk gestegen.

Het eerste dat opvalt is dat jullie onze aanbeveling over de component-indeling hebben opgevolgd. Jullie hebben hem zelfs letterlijk opgevolgd, want de componenten heten nu “frontend” en “backend”. Als jullie functionele namen hadden verzonnen was dat ook prima geweest, zolang de naamgeving maar beter beschrijvend en minder misleidend is dan de oude “src” en “web”.

Ook bij Unit Size zijn onze aanbevelingen gevolgd, het aantal (te) grote methodes is nu een stuk kleiner dan voorheen. De deelscore voor Unit Size is nu drie sterren, wat betekent dat jullie nu op alle door ons gemeten aspecten gemiddeld of bovengemiddeld scoren.

Tot slot is het goed om te zien dat naast deze aanpassingen in de productiecode de hoeveelheid testcode ook is toegenomen.

Uit deze observaties kunnen we concluderen dat de aanbevelingen van de vorige evaluatie goed zijn meegenomen in het ontwikkeltraject.