Delft University of Technology

Software project
CSE2000

Project Plan

Group 3C:
Alexandru Lungu
Luca Becheanu
Alex Ciurba
Melle Schoenmaker
Toma Zamfirescu

TU Coach: *Alan Hanjalic*
TA: *Alves Marinov*
Client: *Educational Innovation Projects*

April 30, 2021

Contents

1 Problem analysis

1.1 Context

TU Delft is one of the most prestigious universities in Europe and its growth in the last years cannot be overlooked. The Bachelor of Computer Science and Engineering (CSE) at TU Delft improved tremendously in recent years. In 2018, it officially became a programme in English, and now the competition to get in is stronger than it ever was before. Quality of education is a priority for the staff, and that is why a lot of software is used to improve the students' experience.
To automate and improve the educational methods, the Educational Innovation Projects (EIP) team developed some platforms that are used extensively by students from all over TU Delft. To name a few: Queue is a platform that allows students to ask questions during lab sessions, Peer is a platform that allows students to do peer reviewing, and TAM is a platform that facilitates the recruiting and scheduling of teaching assistants.
The Bachelor of Computer Science and Engineering at TU Delft has multiple mandatory programming projects, for the courses CSE1105[1], CSE2115[2], and CSE2000[3]. While the projects are extremely well structured and the students' experience has always been positive, grading is a slight problem for such projects. Right now, teachers, head TAs and teaching assistants have no platform that eases the process of grading for such big projects, which makes their job much more difficult.

1.2 Problem Statement

1.2.1 Data spread over multiple files, repositories

When assessing a student for one of the big projects, there is a lot of data that needs to be taken into account: weekly rubrics, attendance, personal issues, the contribution to the project, and notes by TAs or professors to name a few. Everything is in different files or repositories, so accessing it and updating it is not easy. It can sometimes happen that crucial information is lost during the process, so not only the final grading is complicated, but also the whole process.

1.2.2 The need to manually combine the data

At the end of the projects, TAs and professors need to compute a final grade for a student based on all the data they collected during the quarter. To do that, they need to manually combine all the data and come to a final grade. This can take a huge amount of time and it is also prone to error.

1.3 Project Goals

1.3.1 Develop Gradinator

The goal is to develop an application to ease and automate the process of grading for project-based courses within the CSE Bachelor at TU Delft. To do that, we will build an application that keeps all the relevant data together and has all the functionalities so that the effort during grading is minimal.

1.3.2 Have the product usable for the next academic year (2021-2022)

Our goal is to have our product usable for the project-based courses during the 2021-2022 academic year, which starts around the 1st of September.

1.3.3 Build a scalable application

Since TU Delft is constantly growing, we want to build a scalable application. Also, since courses' grading approach changes frequently, additional functionality should be easily implemented into the platform.

1.4 Research on existing products

Gradinator is part of the software products that are identified as Learning Management Systems (LMS). In this section we will define what LMS stands for and compare products that our project is related to, listing their similarities and differences to show the need of Gradinator.

[1] Object Oriented Programming Project
[2] Software Engineering Methods
[3] Software Project

1.4.1 Learning Management Systems (LMS)

Learning management system (LMS) is considered as an important means of knowledge acquisition and learning management in the digital era. As users could be seen as the key stakeholders who impact the system's survival, their attitudes toward system are put in high consideration [1]. Its main purposes are administration, documentation, tracking, reporting, automation and delivery of educational courses, training programs, or learning and development programs. At the moment, the faculty is using a third party product, Brightspace, in which they track course progress, input lectures, exercises and grades that will be more easily evaluated in our project. The process for grading the project based courses is currently difficult and contains a lot input from different sources as specified above, thus our goal would be make this simpler to complete the Brightspace grading.

1.4.2 Brightspace

Similarities. Brightspace is a third-party product adopted by TU Delft. It is a Learning Management System (LMS) that allows teachers and instructors to create content pages for their courses where they can add assignments, lectures, create groups, any much more.
Differences. Grading project-based courses is difficult on Brightspace input because the assessment is more complex than a simple grading scheme - it consists of multiple inputs from multiple sources like TA notes, rubrics, GitLab statistics on commits, lines of code. Brightspace cannot take all of this into consideration as it is has a simple, generalized grading scheme. Thus, for project-based courses, it requires a lot of manual sorting and looking through multiple sources to finally end up with the grades.

1.4.3 Weblab

Similarities. Within Weblab teachers can create pages for their courses. In these pages they can setup coding assignments that can be auto graded based on how well the auto testing went on the code wrote by the students. Assignments can be manually graded based on a rubric which is really similar to the approach in project based courses but falls short for reasons stated below.
Differences. Weblab is not a good example for what our project should be closely related to. Weblab is mostly built for courses that have set assignments that can be done in one or more specific ways, or open questions that can be checked easily, but for us that would not really help because the procedure on how everything is done is way more important in the grading process of these courses (mostly CSE1105[4]) and Weblab does not help us in regard to that.

1.5 Use Case Analysis

Use case analysis, the foundation upon which the system will be built, is a technique used to identify the requirements of a system and the information used to both define processes used and classes which will be used [2]. We will be focusing on two user perspectives, namely the teacher/lecturer (TL) and the teaching assistant (TA), since head teaching assistants (HTA) will be having joint permissions with the former two. We decided on using this process since we want to design our system from the user's perspective in order for it to be complete and reach our final goal of satisfying the user. These Use Cases can be found in Appendix B.

2 Requirements

2.1 Requirements Elicitation

The focus of our project is to create an easy to use and intuitive application which works in a similar fashion to other tools that teaching assistants and teachers already use. To get better insight into what teachers and TAs need or want to see in the application, a survey is distributed to those that are involved in grading courses such as CSE1105, CSE2115[5] and CSE2000[6]. Initially we improved our view on the issues by talking to the client and the TA managing our group, as they can offer us insight in the problems that occur behind the scenes. In the meeting with our client we looked to get a better view on the specific reasons behind the requirements. Using this knowledge it becomes easier to work out questions for the survey.

[4]Object Oriented Programming Project
[5]Software Engineering Methods
[6]Software Project

2.2 Crowd-Based Requirements Engineering

Since we do not have experience in grading project-based courses, we decided to use a technique called Crowd-Based Requirements Engineering (CrowdRE). This concerns the involvement of some of the stakeholders in the requirements engineering process. Our goal is to develop the application in the benefit of the stakeholders, so their opinion on the requirements and prioritization is of utmost importance.

2.2.1 Approach

In order for us to deliver the best possible product, we decided to create a survey in which we got insights from various stakeholders. Since different stakeholders have different tasks during the grading process, we designed the survey differently for TAs and teachers/head TAs. Since we have thought about most of the requirements beforehand, we asked some specific questions, but also added some open-ended questions, to give the stakeholders the chance to come up with new ideas.
All the stakeholders were asked how difficult keeping track of problematic students is. We gave our potential solution and asked if they see any oversight in it. We also included a question where we listed our main requirements, asking how much easier the grading process would become if all those features were implemented in our app. Also, we had a common open-question asking what other ideas they might have that would improve our application.
As separate questions, we asked TAs was how difficult filling in the rubrics is, giving them our potential solution and asking for advice on it. For teachers/head TAs, we wanted to know how difficult computing the final grade is and what would be their preferred way of doing that.
At the end, we thought it would be a good idea to let the stakeholders choose the name of our application, so we gave them some ideas, while also being open to new suggestions.

2.2.2 Results

In total, we received 11 responses, 5 from TAs and 6 from teachers/head TAs. We feel like the survey provided us with a better overview of the problems and we realised that most people involved in the grading of project-based courses had roughly similar opinions about the current process. Also, the wide majority of the people answering the survey had a very good opinion on our approach for the development of the application. Some observations about the answers of the survey:
The TAs answered that filling in the rubrics is rather difficult, but that keeping track of the problematic students is not a major difficulty in their grading process, since they do not have that many students to look after. On the other hand, teachers/head TAs answered that keeping track of problematic students is difficult, since they need to manually look over all of the groups and students.
Teachers/head TAs answered that the task of computing the final grade is not as difficult as the process of gathering all the data during a project, since they only need to do an overview of a student/group. Regarding computing the grade, most teachers/head TAs answered that they would like the option to see an overall grade for a group, and then adjust it for every student.
We compared the results and, after discussing suggestions from the form with the client, we updated the list of requirements and improved our approach towards developing the product.
For a more detailed overview of the survey and its results, please visit the appendix.

2.3 MoSCoW methodology

We've chosen to use the MoSCoW method [3] of prioritizing tasks as it allows an intuitive way of knowing which tasks need to be finished as early as possible to extend the platform. The list of must haves contains all requirements that are necessary for the end product to be usable in the way it is intended to work. The should haves contain elements that are not as significant as the must haves, but are still useful parts of the application that improve user friendliness. Could haves contain 'luxury' additions that would not add large functional capabilities to the application, rather they add cosmetic value to the application. Won't haves are potential parts of the application that are not going to be implemented, because there is either a better alternative or they serve no purpose in the application.

3 Feasibility study

After setting the requirements for the project, we conducted research to evaluate the feasibility of our product under the given conditions. We divided our research into the following sections:

3.1 Technical Feasibility

During the first week of the project, we have already decided on what technology and tools we will use during the development. We do not need additional hardware apart from our workstations, since we are developing a web application. Regarding software, we will use the PHP-based Laravel framework, but also MySQL and JavaScript. Each of these tools are freely available, and the technical skills of our group are good enough, since we are Computer Science and Engineering BSc. students, with plenty of experience in programming. Thus, the project we are developing is technically feasible.

3.2 Operational Feasibility

We need to implement an application that integrates all data from grading into one. Since there are no other teams working on the project and we do not depend on functionalities from other sources, we believe that the application is operationally feasible.

3.3 Time Feasibility

An important factor to consider when conducting the feasibility study is the time it would take us to finish the project. Everyone should work for 40 hours a week, which we consider to be enough given the deadline imposed by the Software Project (7 weeks). To make sure we meet this deadline, we divided our work according to the MoSCoW method, which should guarantee that we at least implement the features that are must haves.

3.4 Summary of Feasibility

Under the given schedule, the given time constraints and the constraints on our resources, our group believes that the product is technically and operationally feasible.

4 Risk analysis

4.1 Sensitive data

A potential part of the application will be the ability for teachers or head TAs to add notes regarding a student's personal reasons for e.g. contributing less or missing a meeting. It is important that these notes are not shown to anyone that isn't authorized to do so. Limitations should be put on these notes to prevent sensitive data from leaking through users that should not be involved with this data in the first place.

4.2 GDPR compliance

Our project involves collecting a large amount of data about students that are participating in project-based courses. The General Data Protection Regulation (GDPR) imposes sanctions in case the collected personal data is misused or leaked. There are two risks in this case: Firstly, we could have not covered all legal aspects completely, and some data ends up being used incorrectly. The other risk is suffering a breach due to a lack of sufficient cybersecurity or malicious intent of an user.

4.3 Lack of knowledge about PHP

For all members from our group this is the first time coding in PHP, and subsequently, the Laravel framework. This poses a risk, because we are not familiar with these technologies and it might take longer than expected to finish the project.

4.4 Additional requirements

There exists the possibility that new requirements will appear at some point. If this happens, it can interfere with our project plan and cause delays. To minimise this risk, we talked with the client about the requirements thoroughly to make sure that no "surprise" requirements will come up, or at least limit their number.

5 Project approach

5.1 Backend: PHP/Laravel

The backend will be implemented in PHP with the help of the Laravel framework that allows us to provide a REST API to the client. This decision was made because the EIP is already using it for a number of their apps. We also wanted to use something different than Spring, as we have used Spring for the last two projects and want to learn how to use other frameworks. Routing with Laravel is very intuitive, and Blade is a simple yet effective way to manage views. A final reason we chose Laravel is that querying the database is way easier using Laravel than with Spring, thanks to Eloquent.

5.2 Database: MySQL

MySQL is one of the most popular database systems for web-based applications. It is free but it is updated regularly with more features and security improvements. We chose it because in our opinion, it is the standard and it is very reliable while also providing comprehensive support for every application development need. It does not need that much setup to get going and its performance is high enough to be used as a high-speed transactional processing system or a high-volume web site that services a billion queries a day, MySQL can meet the most demanding performance expectations of any system [4].

5.3 Frontend: Bootstrap/NodeJs/Vue

Our current plan is to use Bootstrap for frontend since it is a lightweight and consistent framework that will satisfy our needs, help us save time and adhere to the best practices. If we have enough time we can try to utilise more advanced frameworks such as Vue, but that is not necessarily needed and it would only be used to have a smoother look for the interface.

6 Development process

We will be using Agile software development [5] since it is does not have a rigid structure and encourages flexible responses to change. We will make use of weekly sprints from the Scrum [6] framework where we will hold one big meeting every Monday morning in which we will structure our work week together. During the week we plan to have 3 shorter 30-minute meetings such that each member of the group explains the state of the work done the previous day and what needs to be done on the current day. Finally, we will have one big meeting at the end of the week where we discuss our progress, do the sprint review and sprint retrospective, merge requests and discuss any other issues we encountered.

During the TA/Coach/Client meetings we will have a designated chairman who will ask the questions and present demos, and 2 note takers responsible for writing down every important thing that's discussed.

We will make use of tools such as:

- Gitlab: The main use of Gitlab is its version control system, however we will also use this tool for: checking quality of the code through merge requests which will require at least 2 other developers to approve it after they looked at the code and confirmed it works as intended; keeping track of task through the usage of the boards feature, which allows us to see which tasks are currently being done and which are already finished, and also assigning each task to the developer(s) that are doing them.
- Discord: This will be our main form of communication between team members, and also where our meetings will take place, since currently, due to the coronavirus pandemic, all lectures, meetings and presentations have been shifted to be done digitally. Moreover, our team is made of international students and, because the learning environment is online, most of us are not in the Netherlands to be able to schedule in-person meetings.
- Whatsapp: We will use Whatsapp for communication when someone is unavailable on Discord.
- Mattermost: This tool will be used to communicate with our client and TA about our progress and any questions related to the project.
- Jitsi: We will use Jitsi for our meetings with the TA.
- Microsoft Teams: Microsoft Teams will be used for our meetings with the client.

References

[1] Nhu-Ty Nguyen. A study on satisfaction of users towards learning management system at international university – vietnam national university hcmc. *Asia Pacific Management Review*, 2021.

[2] Daryl Kulak and Eamonn Guiney. *Use cases: requirements in context.* Addison-Wesley, 2012.

[3] *A Guide to the Business Analysis Body of Knowledge (2 ed.)*, chapter Requirements analysis, MoSCoW Analysis. Section 6.1.5.2. International Institute of Business Analysis, 2009.

[4] K. Nadeem. Online shopping portal.

[5] Carlos Tam, Eduardo Jóia da Costa Moura, Tiago Oliveira, and João Varajão. The factors influencing the success of on-going agile software development projects. *International Journal of Project Management*, 38(3):165–176, 2020.

[6] Marcelo Morandini, Thiago Adriano Coleti, Edson Oliveira, and Pedro Luiz Pizzigatti Corrêa. Considerations about the efficiency and sufficiency of the utilization of the scrum methodology: A survey for analyzing results for development teams. *Computer Science Review*, 39:100314, 2021.

A Survey

A.1 Questions

The following questions were asked depending on the person answering being a head TA/teacher or a teaching assistant.

A.1.1 Teaching assistant questions

- What is your personal view on the current way of filling in rubrics? (Getting rubric templates, filling in the file, uploading it to a repository)
  -MCQ that contained open question to support the answer from this question.
- To solve the problem of filling in rubrics and storing them in repositories, we thought of the following solution. For every week of the project there is a checklist table built into the application which allows you to fill in and save the rubric. Are there any oversights in this solution, what would you change about it?
  -Open question
- Sometimes students stand out due to low scores in BuddyCheck or what comes up during meetings. What is your view on the current method of keeping track of problematic students?
  -MCQ that contained open question to support the answer from this question.
- To prevent you from having to send emails to head TAs and teachers about problematic students we thought of the following solution. You would have button to add problematic students to a list which is visible to you, head TAs and teachers, with a note explaining the problem. Head TAs and teachers would be notified about the problematic student inside the application and can take appropriate measures. Are there any oversights in this solution, what would you change about it?
  -Open question
- After having options to do what is listed in figure 1, how easy would the grading process become, compared to the current method.
  -MCQ that contained open question to support the answer from this question.
- Apart from the ideas we came up with, what else would you want to see in the application that would ease the process of grading?
  -Open question

A.1.2 Head TA/teacher questions

- Sometimes students stand out due to low scores in BuddyCheck or what comes up during meetings. What is your view on the current method of keeping track of problematic students? (TAs notice less participation, they notify you through an email/message. Or they have consistently low BuddyCheck scores)
  -MCQ that contained open question to support the answer from this question.
- To keep issues regarding problematic students inside the application we've thought of the following solution. We would give you and TAs a button to add problematic students to a list which is visible to you, with a note explaining the problem. You could then create interventions for this student (such as deadline extensions/extra work). Are there any oversights in this solution, what would you change about it?
  -Open question
- What is your personal view towards the current way of computing grades? (Going through rubrics, looking at attendance, checking git analysis, then combining all factors into a grade)
  -MCQ that contained open question to support the answer from this question.
- How would you like the final grade to be computed? Please use the last question of this section for suggestions. You can choose multiple answers.
  -MCQ that contained 4 answers
  - Automatically give intermediate grades for different components. (rubrics, attendance, git anaylsis, etc.)

  - Show a penalty based on the student's attendance/failed deadlines.
  - First give a grade for the group, then change it upwards/downwards for individual group members. (based on attendance, git analysis, etc.)
  - Show an overview of the student's activity and then manually register the grade.

- After having options to do what is listed in figure 2, how easy would the grading process become, compared to the current method.
  -MCQ that contained open question to support the answer from this question.

- Apart from the ideas we came up with, what else would you want to see in the application that would ease the process of grading?
  -Open question

A.2 Responses

We received responses from five teaching assistants and six head TAs or teachers. A summary of the responses for both of these groups can be found below. Answers correspond to the respective bullet point in the Questions section above.

A.2.1 Teaching assistant responses

- On average TAs found that filling in rubrics is currently slightly difficult, rating it at 2,75. Some problems that were noted were: Spending a lot of time retrieving the correct template, updates to the rubrics didn't always find a way back to the pdf or js files the TAs were using, and quite a lot of work is manual despite it being automatable.

- Given our solution proposal for rubrics the TAs were positive. Some remarks were to ensure that head TAs and teachers can still update the rubrics, that intermediate grades should be exportable, that there is a need for open fields (either per week or per specific part of the rubric), and that parts of the rubric could be entered automatically based on git analysis and other git tools.

- For keeping track of problematic students in the current system, the average rating was a 3,25. The explanations that were given came down to it being quite easy for the TAs to notice low BuddyCheck scores and writing them down in their notes. They could then notify the head TAs or teachers about the problems and no longer have to worry about it. There were two suggestions made: Give an overview of the history in BuddyCheck results, so that the TA can easily see if a student that was lacking has improved their behaviour, and to give a way to message/notify the teachers about problems in groups in the application, so that problems are not lost in heaps of messages/emails.

- Given our solution proposal for problematic students the response from 50% was that it offered no clear benefits to using Mattermost, the other part was positive. An interesting suggestion was made to have traffic light levels for problematic students: red being extremely problematic, yellow being slightly problematic and green being good. TAs would then be able to assess their students on this scale on a weekly basis and in the case of a problem, they would be able to give further explanations. Head TAs or teachers would be notified about these problems per group.

- After giving the functional requirements listed in figure 1, the average response was a 4,25. A key improvement that was noted was that having an easy way to register attendance would be great.

- The suggestions that were given covered the following points: adding links to each groups info (repository, overleaf), and showing history on some statistics (using graphs).

A.2.2 Head TA/teacher responses

- For keeping track of the problematic students, the average answer was that it is quite difficult to keep track of these students, adding to the question that their problems with regard to this issue were that sometimes the problem got to them too late because they couldn't detect it fast enough or that the course has a Gitlab repository where they have information but is hard to maintain.

- To our proposed solution, the people that answered added a lot of suggestions, such as: if the student failed the course, you should be able to remove them from a group, have the system indicate all problematic students in an ordered list of most 'problematic' to least 'problematic' (with a way to deactivate problems once they are solved), it would be better to have notifications at the group level so that the head TA or teacher can look into group related issues more easily, and finally TAs should be able to add relevant information from e.g. BuddyCheck to the problem.
- In regard to the current way of computing grades, the average answer was that it is easy but it becomes problematic when multiple people are involved in the grading process so a feature that adds some automation in the merging of different components would be great.
- As for the preferred way for the grade to be generated or computed five out of six answers wanted to first grade the group, then alter this grade from person to person. Intermediate grades based on activity, rubrics and attendance were also an appreciated suggestion. Finally an option that needs to be considered is to give an overview of the students activity to prevent the person grading from having to look all over the place.
- After giving the functional requirements listed in figure 2, the average response was a 3,83. One suggestions was to let the system flag good/bad students so that the person grading can compensate for it in the grade. Further notes contained that having everything in one place would be a good improvement.
- The suggestions from head TAs and teachers were: have a place to write down notes that are visible to other teachers or head TAs, an overview of problem cases, integrations with MyTudelft and Brightspace for grade exporting and a way to export and share feedback in rubrics with students, and lastly personal evaluation or self proposed grades from students.

B Use cases

Hereby is the list of use cases mentioned in section 1.5.

B.1 Teacher/lecturer (TL) use cases

- TL wants to log into the system with their NetID using Single Sign On.
- TL wants to register attendance or notes per meeting, per group, and also on an individual level.
- TL wants to add interventions, such as deadline extensions or doing extra work, and then follow up on whether students lived up to these interventions by receiving a notification after the deadline.
- TL wants to see all groups, and have a list of the events within a group.
- TL wants to upload reports from BuddyCheck and git analysis.
- TL wants to create, edit or import rubrics for a course edition, these rubrics should apply to every group taking part in the course edition.
- TL wants to mark notes regarding personal reasons as confidential to prevent anyone unauthorized from seeing them.
- TL wants to post deadlines for TAs during the project.
- TL wants to filter the list of groups to find a group or a specific student.
- TL wants to export grades per group/per course from the application and import them to Brightspace/Osiris.
- TL wants to see grade statistics per course edition.
- TL wants to import rubrics as CSV[7] files instead of having to create rubrics inside the application.
- TL wants to view the percentage of students that have received their grade.
- TL wants to import a list of students that are each linked to a group.

[7] https://en.wikipedia.org/wiki/Comma-separated_values

B.2 Teaching assistant (TA) use cases

- TA wants to log into the system with their NetID using Single Sign On,
- TA wants to register attendance or notes per meeting, per group, and also on an individual level.
- TA wants to receive a notification that deadlines for student-specific agreements passed.
- TA wants to log the problems of the group or an individual.
- TA wants to receive a notification if a specific student stands out in BuddyCheck reviews.
- TA wants to see the deadlines imposed by teachers.
- TA wants to filter his list of groups to find a group or a specific student.
- TA wants to see grade statistics per course edition.
- TA wants to view the percentage of students (in the groups they manage) that has received their grade.

C Requirements

As a general note each user will have a role that gives access to that part of the application, marked as follows:

- Teaching assistant (marked TA)
- Head teaching assistant (marked HTA)
- Teacher/lecturer (marked TL)

C.1 Must haves

- Ability for a user to register attendance or notes per meeting, per group, and also on an individual level. (TA, HTA, TL)
- Ability for a user to enter interventions, such as deadline extensions or doing extra work (HTA, TL).
- Users must be able to follow up on whether students lived up to these interventions by receiving a notification after the deadline. Should allow for multiple follow ups, similar to threads. (HTA, TL)
- A way of notifying a user that deadlines for student-specific agreements passed. (TA, HTA, TL)
- A user must be able to see all groups, and have a list of the events within a group. (TL)
- Dummy login authentication must be implemented, as a placeholder for when SSO can be integrated.
- Ability to create, edit or import rubrics for a course edition, these rubrics should apply to every group taking part in the course edition. (HTA, TL)
- Ability to import a list of students that are each linked to a group. TAs must then be assigned to groups by the user. (TA, HTA)
- The hierarchy order in the GUI must be as follows: Course, Course Edition, Groups, Weeks, [Rubrics, Attendance, Notes, Interventions] (TA, HTA, TL)
  For TA: Interventions are not shown to all TAs, only the TA responsible for the group unless the HTA/TL marked it as private. Only groups that belong to that TA are visible. For HTA and TL: Interventions are visible. All groups are visible.

C.2 Should haves

- A user should have a weekly 'problem signal' menu which allows them to fill in problem levels for each student. Problem levels are defined as some value (green, yellow, red or numbers from 0 - 5); the TA can decide how problematic certain students are. (TA, HTA, TL)
- A user should be able to mark notes regarding personal reasons as confidential to prevent anyone unauthorized from seeing them. (HTA, TL)
- A user should be notified if a specific student stands out in BuddyCheck reviews. (TA)
- A user should be able to post deadlines for TAs during the project. (TL)
- The interface could contain an agenda which lists the deadlines from interventions/teachers/assignments for the current day, coming days, and previous days. (TA, HTA, TL)
- A user should be able to filter within the list of groups to find a group by a specific student (on student number/email/name) or on TA (for HTA and TL). (TA, HTA, TL)
- Ability to upload reports from BuddyCheck and git analysis. (HTA, TL)
- Allow user to add important links/integrations for each group. (Gitlab repository, Overleaf) (TA, HTA, TL)

C.3 Could haves

- The grades per group/per course could be exported from the application and then imported to Brightspace/Osiris by the responsible user. (HTA, TL)
- The interface could allow the user to change their color scheme by picking from a list of color schemes (Light, Dark, etc.). (TA, HTA, TL)
- The user could see grade statistics per course edition. (TA, HTA, TL)
- The user could be able to import rubrics as CSV files instead of having to create rubrics inside the application. (HTA, TL)
- The user could be able to view the percentage of students that has received their grade (TA, HTA, TL)
  For TA: Show the percentage of students with a grade in the groups they manage.
  For HTA and TL: Show the percentage of students with a grade over all groups.
- Allow user to deactivate problem cases after they have been resolved. (HTA, TL)

C.4 Won't haves

- The final system won't have it's own authentication system, rather as listed in the must haves, the system will redirect to SSO.

C.5 Non-Functional

- The system backend shall be implemented in PHP using the Laravel framework.
- The system shall use MySQL databases for the data storage back-end.
- The system shall use Bootstrap/NodeJS as JavaScript frameworks for the front-end.
- The system shall use Gitlab for CI/CD and version management.
- The system must cope with usage of teams of (20+) TAs and teachers that monitor courses of approx. 450-500 students.
- The system shall have at least 85% meaningful branch test coverage.
- The application must be working on all major browsers such as Google Chrome, Mozilla Firefox, Opera or Safari.

Figures

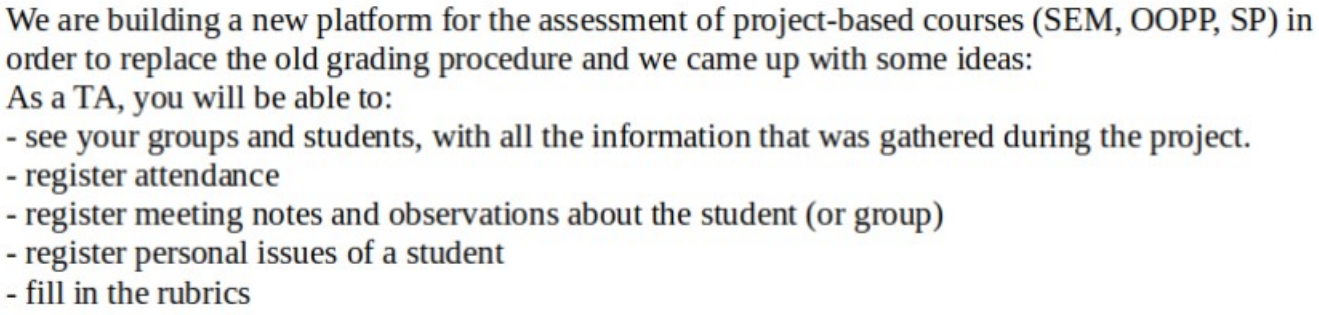

Figure 1

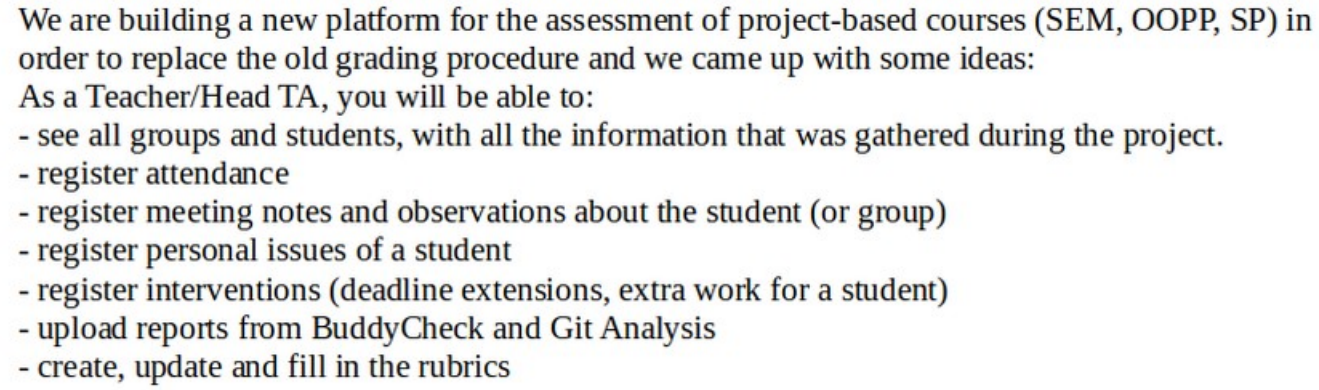

Figure 2