

August Lund Eilertsen

Team Accelerator

Web-application for monitoring team performance and progression in project related academic work.

Master's thesis in Computer Science, Software Engineering

Supervisor: George Adrian Stoica

May 2020

August Lund Eilertsen

Team Accelerator

Web-application for monitoring team performance and progression in project related academic work.

Master's thesis in Computer Science, Software Engineering
Supervisor: George Adrian Stoica
May 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science



Contents

1	Introduction	5
1.1	Introduction	5
1.2	Method and approach	5
1.3	An important notice	6
1.4	Summary	6
2	Background & Research	6
2.1	Challenges with teamwork	6
2.2	Tactics to overcome challenges for group members	7
2.3	Tactics to overcome challenges for the instructor	7
2.4	Other solutions	8
2.5	Student survey	8
2.5.1	Survey results	9
2.6	Interviews	11
2.6.1	Interviews with professors	11
2.6.2	Interviews with teaching assistants	12
2.7	A need for a group status/progression application?	12
2.7.1	Research conclusion	12
3	Design	13
3.1	Methodology	13
3.2	Design of the graphical user interface	13
3.2.1	Initial design	13
3.2.2	Design iteration 1	14
3.2.3	Design iteration 2	15
3.2.4	Design iteration 3	17
4	Architecture	20
4.1	Overall Architecture	20
4.2	Frontend Architecture	21
4.3	Backend Architecture	22
4.4	Database Architecture	22
4.5	Feide user and authentication	23
5	Implementation	25
5.1	Implementation method	25
5.2	Choice of technology	26
5.2.1	Frontend: React & Redux	26
5.2.2	Backend: Django REST Framework	26
5.2.3	Database: PostgreSQL	26
5.3	Code and project structure	27
5.3.1	Backend	27
5.3.2	Backend folder & app structure	31
5.3.3	Frontend folder and & component structure	31
5.3.4	Frontend React code	32
5.4	Hosting the application for testing with Heroku and Surge	35
5.4.1	Heroku for backend	35
5.4.2	Surge for frontend	35

5.5	Hosting the application for production	35
5.6	Handling the course permissions	36
5.6.1	First solution	36
5.6.2	Second solution	36
5.6.3	Current solution	36
6	Privacy Policy and GDPR	39
6.1	GDPR Compliance	39
6.2	NSD	39
6.3	Privacy Policy In-App	40
7	Results and future work	41
7.1	The application	41
7.1.1	The registration process	41
7.1.2	Student in-app functionality	42
7.1.3	Staff in-app functionality	46
7.2	Django backend admin panel	51
7.3	Qualitative user testing	54
7.3.1	Test form	54
7.3.2	Introduction for student test	55
7.3.3	Student test tasks	55
7.3.4	Introduction for staff test	56
7.3.5	Staff test tasks	56
7.4	Qualitative user testing results	57
7.4.1	Student tests	57
7.4.2	Staff tests	59
7.4.3	Elements changed based on the user testing	62
7.4.4	Elements that can or should be changed based on the user testing	62
7.5	Quantitative user testing	63
7.6	Conclusion & future work	63
8	Resources used	64

Appendices

A The old login process

B Additional and alternative design

C Personas

List of Figures

1	Results of survey question 1	9
2	Results of survey question 3	10
3	Results of survey question 4	11
4	Initial design of the student side of the application	14
5	Design iteration 1	15
6	Design iteration 2 - Student side	16
7	Design iteration 2 - Professor/Teaching assistant side	17

8	Design iteration 3 - Student after-rating page	18
9	Design iteration 3 - Professor/Teaching assistant side	19
10	Overall architecture	20
11	React & Redux architecture - simplified	
12	The backend architecture - simplified	22
13	Database architecture	23
14	The Team model	27
15	Endpoint for getting the user	28
16	Format of the JSON returned	29
17	The SubjectSerializer	29
18	The UserSerializer	30
19	Screenshot of a simple POST request	30
20	Folder structure of backend project	31
21	Frontend folder structure	32
22	TeamList component	33
23	TeamList component	34
24	Usage of TeamList component	35
25	Access control upload flow	38
26	Delete user in app	39
27	Privacy policy in app	40
28	Login process	41
29	The rating side for students	42
30	Student side	43
31	Student profile page	44
32	Changing to another course	45
33	Staff side of the application	46
34	Continuation of staff side of the application	47
35	Continuation of staff side of the application	48
36	Continuation of staff side of the application	49
37	Pin and unpin teams	50
38	Staff profile	51
39	Admin panel, main overview	52
40	Admin panel, user overview	53
41	Admin panel, view and change one user	54
42	Result student 1	57
43	Result student 2	58
44	Result student 3	58
45	Result student 4	59
46	Result staff 1	59
47	Result staff 2	60
48	Result staff 3	61
49	Result staff 4	61
50	Appendix: Login process	
51	Appendix: Additional design of student side	
52	Appendix: Additional design of staff side	

Frontend code repository

https://gitlab.stud.idi.ntnu.no/augustle/master_react

Backend code repository

https://gitlab.stud.idi.ntnu.no/augustle/master_api

Testing the application

If you want to test this application, contact one of the following emails, and you will be given a test user:

- augustle.lund@gmail.com
- stoica@ntnu.no

1 Introduction

1.1 Introduction

If we take a look at the evolution and how the human kind has evolved over thousands of years, we can conclude with one major important factor: If our ancestors were unable to work in teams and get along in groups, they would probably not survive [8]. We are a result of the people who had the ability to cooperate with others and make decisions as a group, rather than lone wolves. In this master thesis, we are going to discuss the many important factors of teamwork and how to succeed with it. It will cover research on important factors on how to succeed with team work. It will also take the reader through all the stages of development of a group status monitoring application. The results from testing the application with various methods will be discussed.

The overall goal of the project is to assess and visualize the status of group projects, and construct a web application that provide help to teams that struggles with their work and performance. The research will serve as a justification on why there would be beneficial to have a software to monitor and control the overall status of teams in university courses. The development of this product will also be based on interviews with instructors and teaching assistants at NTNU, and a student survey.

The final product is a web application for monitoring and assessing team status. This web application is tailor-made for smartphones, but also works on desktop. Team status in this scenario means how well a team is doing based on feedback from students. It is a twofold application consisting of two different user interfaces. The first user interface is for students and allows them to rate their group on a weekly basis, with a number between 1 to 5 respectively very poor to very good. The average score will be calculated and displayed to the students and the professor. The second user interface is for professors and teaching assistants. The instructor and TA will have the ability to view all the different groups and their average score including individual scores.

1.2 Method and approach

The software development method used in this thesis is agile development. For the overall structure and tasks, meetings and retrospectives was held every two weeks with the project supervisor, with exceptions. The technical development part included continuous integration with agile development. The problem was divided into many small tasks, and each tasks was solved in incremental cycles. The delivery includes a study of research on important key elements to succeed with teamwork, the graphical design of the application, the architecture of the web-application, the actual implementation and development of the application and the results of it. The delivery also includes the results of testing the application on the relevant target audiences. Other elements used in the method and approach:

- Design research and user centered design was applied
- Personas and a number of empirical research methods was applied, such as interviews and user tests.
- Prototypes was used over several iterations, to test and verify the product.

”Group work” and ”teamwork” are used interchangeably in this thesis, and means the same thing.

1.3 An important notice

Due to the outbreak of the COVID-19 virus during the spring semester of 2020, the results of testing this application were weakened. The original plan was to test the app on various students and teaching assistants in person. This had to be adjusted, and all the user-testing was performed virtually. The application was also planned to be presented in one of the lectures of TDT4180 - Human Computer Interaction, but this had to be canceled. The result of this is fewer test results, but sufficient.

1.4 Summary

This master thesis is about design, development and testing of a new web application that will focus on problems related to academic team work, and how such problems can be solved more easily with the use of such an application. It also covers the underlying research and theories related to team work, problems related to team work, and how to succeed with it.

From start to end, the reader will be guided through the whole process of the construction of a web application based on the research and theory part of the thesis. The reader will then be guided through the design section where the major decisions on how the application was going to look were taken, included the first design sketches. After this, the architecture section is presented, where the reader will gain an insight of the underlying technical parts that is the basis of the application, both on the frontend and backend side. After this, the reader will gain an insight in how the application actually was developed through the implementation section that describes the code structure, the technologies used, programming languages and more. Finally, the reader will be presented the results, which includes the final application product, and the results of the user testing of the application in various settings.

2 Background & Research

2.1 Challenges with teamwork

Most people have one or several experiences with group work that did not work out well. Maybe the group was a mismatch from the start, or the group developed bad norms, conflicts and unequal workload distribution over time. Anyways, the final product was likely a poor-quality product. Newly formed groups are never the same, as each individual is unique. But no matter how many different personalities there are in a group, it is critical to find common ground right from the start, without misunderstandings. We are going to discuss several different tactics on how to succeed with a team, despite the differences that might exist between the individuals.

The most common problem reported in group work is uneven workload, also called “free riding”. Another very common problem is lack of group norms and bad communication [10]. Often in groups that perform bad, these things were not established and clarified right from the start. Such bad habits are easy to develop. The result is a lack of cohesion and integration. These things can often lead to conflicts, which makes the situation even worse, and if the team did not establish any norms on how to solve conflicts beforehand, it worsens the situation. Other common problems are domineering personalities [2] [9], difficulty getting started due to no development of common ground, roles or norms and inability to focus on the tasks.

2.2 Tactics to overcome challenges for group members

Researchers studying strategy teams at IBM found that the emotional climate and norms established by teams in the first few minutes of their encounters tended to persist and become resistant to changes [4]. It is very important to establish group norms at the very start of the project [7]. This means a set of “rules” and a common ground for the group. Discussing expectations, responsibilities, setting goals and establish basic communication habits/rules [3] [7] [8]. Setting roles is important so that each member has a clear understanding of their own responsibilities, but equally as important is knowing the roles of the other team members [10] [3] [7]. Setting a detailed plan and agenda makes it easier to get the project started [2] [9]. Also, it is important to take time to recognize the other team members accomplishments, either orally or through software. Without recognition, some people can end up feeling unappreciated[1].

The team has to make sure each member has an equal chance to speak, contribute and make suggestions. There is also often an advantage to promote a group-leader, which has the responsibility of scheduling meetings etc [9]. A study made by google which is called “Aristotle”, points out the importance of structure and clarity. Each team-member’s expectations of their tasks, the process of fulfilling these expectations, and the consequences of each member’s performance are important for the effectiveness of the team [6]. Another important tactic to overcome group challenges is to regularly assess the team and how the team is doing. Unhealthy climate is often left to grow and become the norm. It is important that the team and the team-leader reflect on and assess team behaviors and practices [1] [7]. Frequent feedback improves teams performance. Team coordination requires member interaction. More and better interaction improves project outcomes [5]. Setting goals and having a leader who is responsible for overall plans and stakeholder communication is important. Goal-oriented team leadership strongly improves performance [5].

2.3 Tactics to overcome challenges for the instructor

The instructor can do several things to promote a healthy team-environment and an effective team. First of all, the instructor should provide the group with opportunities to discuss their expectations, to set roles, setting group norms and discuss communication norms with the group [10] [2]. The instructor should make sure that the choice of technology promotes collaboration. It is wise for the instructor to provide the students with some kind of tool that makes peer evaluations possible. This gives the instructor an opportunity to identify individual effort and “free-riders” [3]. Studies show that students report greater satisfaction with group work if the instructor has implemented methods to monitor and manage groups and the status of the groups [3].

Another study has shown that there are other methods to reduce “free-riders”, and the most relevant for this project is the option to “divorce” a team member, or the option to leave a team. [3]. Peer-evaluation comments are not useful for identifying group dysfunction. Researchers have recommended peer ratings in form of dividing points between group members, and has been shown to be an effective method to provide an effective team where everyone contributes [3]. The ratings are anonymous, and the students will only see their own points. This insight is especially relevant for this project and thesis.

Either if it is through a software or just spoken out, the instructor should make sure that each student understands the assignments, the purpose of the project, the learning objective and the skills that need to be developed through group work. Also, the instructor needs to help

the students manage and solve conflicts and disagreements [2] [8]. It is worth to mention that conflict is not necessarily negative, but can help to provide the energy a motivation necessary to do things better. In many cases, without conflict, we do not learn. The question is not whether conflict is good or bad for teams - it is whether it is managed so that we can channel its energies effectively [4]. Relationship conflicts harm the performance, while task-related conflicts enhance performance [5].

Conflicts are probably inevitable in teamwork; the question is how to manage them. Teams that focus on conflict management do better, according to a study measuring both team and product performance [5].

A method to help instructors with the challenges involving their teams is to have the students submit weekly progress reports [2] [9]. These reports should include:

- Who attended the meeting
- The objective for next week/meeting
- What the group discussed during the meeting
- The current effectiveness and productivity of the team

It is important that the instructor invites the members of the group to provide honest feedback [1]. Anonymity to some extent may be important in these cases, otherwise the feedback have a likelihood of being dishonest. Sometimes there is no other way than to break up the group, but in most cases it is important to try to avoid breaking up the group. If a breakup happens, the dynamics, habits and norms of the original group will be negatively affected, and the addition of new members to other groups with established norms will likely disrupt their dynamics [2].

2.4 Other solutions

There are a lot of tools supporting teamwork on different aspects, but none of the current solutions I found, solely focuses on the status and progress of the team in terms of effectiveness and performance. The most widely used tool in Norwegian universities is blackboard. While blackboard handles certain content and functionality related to group work, it does not handle monitoring group status and performance.

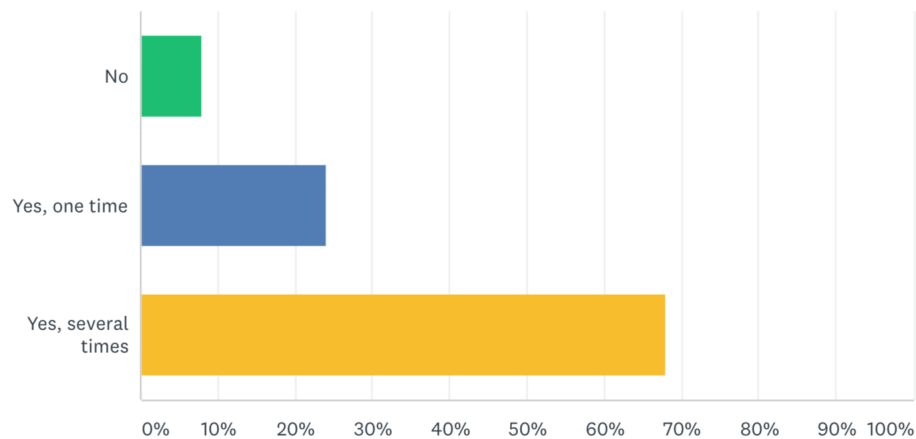
2.5 Student survey

To get approval of the concept and create a broader justification for developing the application, a survey was made and targeted at NTNU students. The three most relevant questions from the survey are displayed below with results.

2.5.1 Survey results

In total there was 50 students that answered the survey. Below are the results of the various questions.

Question 1: Have you experienced working in a bad group that did not work because of "free-riders", bad workload distribution, bad communication etc?

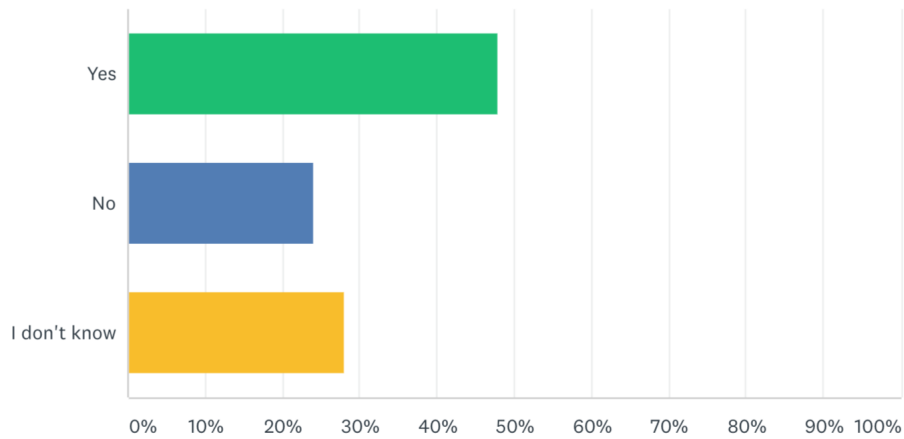


ANSWER CHOICES	RESPONSES
▼ No	8.00% 4
▼ Yes, one time	24.00% 12
▼ Yes, several times	68.00% 34
TOTAL	50

Figure 1: Results of survey question 1

It is clearly a majority of the asked students that has experienced troubles with group work. 68% of the students answered that they have experienced trouble several times with group work while 24% answered one time. This indicates that bad group work occurs a lot, and there is a need for means to deal with this.

Question 3: Could you regularly visit an app on mobile or PC, and anonymously report in on how you feel about your group with a number from 1 to 5, where 5 is very good and 1 is very bad ? The mean score based on every member's answer will be displayed to you, the other group member's and the professors/TA's.

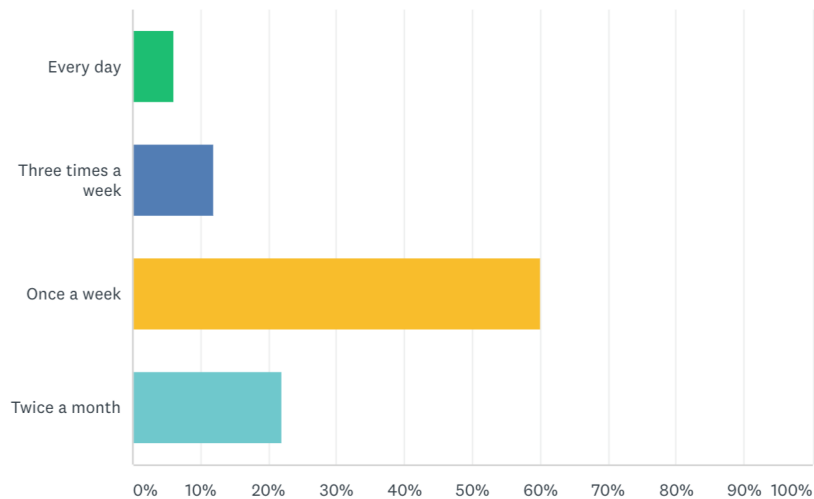


ANSWER CHOICES	RESPONSES	
▼ Yes	48.00%	24
▼ No	24.00%	12
▼ I don't know	28.00%	14
TOTAL		50

Figure 2: Results of survey question 3

When the students were directly asked about the application, 48% answered that they would use such an application, versus 24% that answered the opposite. This is positive, and indicates that an group status monitoring application could be beneficial.

Question 4: How often would you be willing to go into the application and report the status of your group?



ANSWER CHOICES	RESPONSES
Every day	6.00% 3
Three times a week	12.00% 6
Once a week	60.00% 30
Twice a month	22.00% 11
TOTAL	50

Figure 3: Results of survey question 4

The majority of the students with 60% would be willing to visit the application once a week. Based on these results, the product will give the user the possibility to rate the group once a week.

2.6 Interviews

2.6.1 Interviews with professors

There was conducted interviews with two different professors on NTNU. During the interviews, they were presented with a simple prototype of the product. The purpose of these interviews was to get valuable feedback on the design presented, and have them pointing out important key elements for further development.

Interview with professor Professor 1

Professor 1 thought the concept of having an application that focuses on monitoring and im-

proving group status and performance was a useful and good idea. Himself as a professor see the challenges with group work, and he was positively tuned with the concept. During the interview, he pointed out several elements with the design that could be improved, and mentioned several key elements that should be included. The specifics of these elements will be described in the design section.

Interview with professor Professor 2

Professor 2 was positively tuned regarding the concept and design, but his biggest concern was how to make the students go into such an application and give weekly ratings. What motivates them to do so?

2.6.2 Interviews with teaching assistants

The application will potentially be used by teaching assistants as well as professors, therefore it was important to gather feedback from at least one teaching assistant currently managing a project related course at NTNU. One interview was conducted with a teaching assistant, and the results of it will be mentioned under the design section.

2.7 A need for a group status/progression application?

2.7.1 Research conclusion

With the amount of studies done on group work, a lot of the same challenges and the same tactics to solve these challenges are presented. These challenges are bad group norms, bad communication, “free-riding”, unequal workload, avoidance of responsibility, lack of commitment and much more.

The main tactics presented for solving the different challenges are establishing group norms, setting roles and goals, regularly provide feedback to the team, discussing expectations, facilitate peer evaluations, setting a plan and an agenda and much more that involves structure and clarification from the very start. Many of the tactics involve the students providing input in the form of feedback, peer evaluations, peer ratings, discussions etc. If an instructor should have the ability to see any progress status on a group, the students have to provide some input. The goal here is to find an easy, quick and informative way for the students to provide input so that the group status and progression easily can be revealed and done something about if necessary. Following are some possible features of a potential group work application based on the research and the survey:

- **Feedback:** The most important factor would be a method for students to give feedback about what he/she feels about the group. This will be a simple rating from 1-5 that the students have to do once a week. The instructor should have the ability to see the overall rating, see individual ratings, and take certain actions to organize the groups better and get an informative overview.
- **Meeting notifications:** A possibility for the instructor to invite the group to a meeting to solve the problems the group may have.
- **Role assignment:** To give structure and clarity to the team, roles should be assigned and displayed in the application. The research indicates that roles are important.

An application focusing on preventing bad and ineffective team work, and facilitate easy communication between instructors and teams, would be beneficial for students as far as the research states. If this is actually true is yet remained to be proven. The first step is to try out a basic feedback functionality. Every team member simply rates how they feel about the progress and the overall satisfaction of their team every week. The instructor should be able to view all group statuses, individual scores, and certain actions to provide an informative overview. The web application will first and foremost be customized to fit smartphone screens, so that users easily can access it.

3 Design

3.1 Methodology

During the project, an agile way of working was used. There were scheduled meetings every two weeks, with exceptions when meetings was not necessary. These meetings consisted of the current work progress, retrospectives and future work. This working methodology has elements of the known Scrum framework, which is an agile process framework used for teams in software development.

3.2 Design of the graphical user interface

The design was developed in an incremental way as described in the previous section. Only the most important parts of the design will be provided in the main report. The rest of the design can be viewed in the appendix.

3.2.1 Initial design

At the very start of the project, the design was solely based on the project description. With own thoughts and experience with GUI design, the first sketch of the design was made. The first design only included the student side of the application:

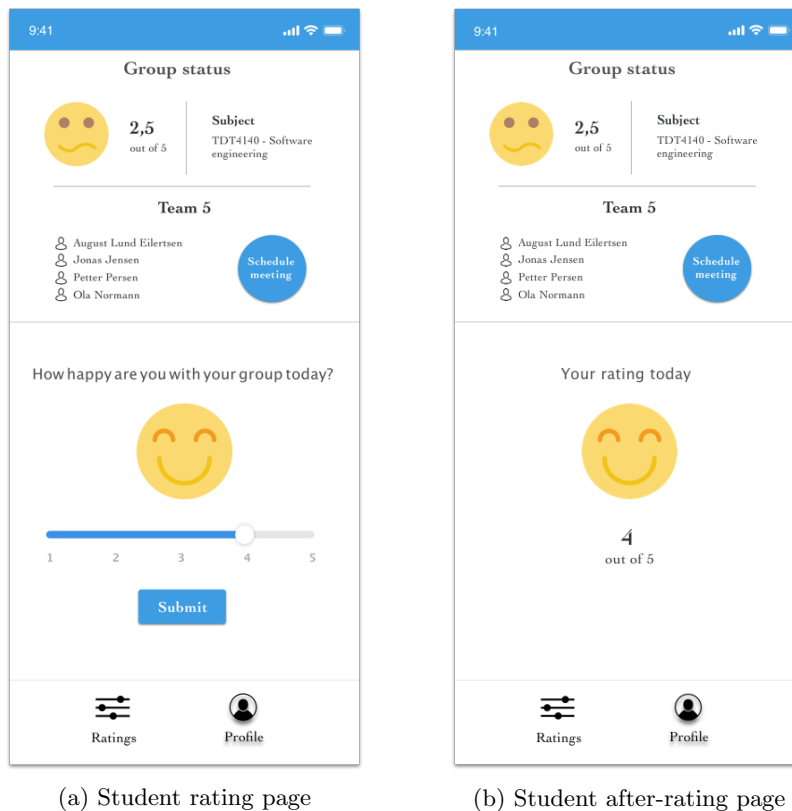


Figure 4: Initial design of the student side of the application

3.2.2 Design iteration 1

After the first sprint and retrospective with the project supervisor, important decisions were made. The following changes were the most important:

- Only having the rating functionality appearing at the start. The display of the overall team score can possibly influence the student's rating.
- After rating, the overview screen should appear
- Rename the "Schedule meeting" button to "Request assistance". This button should make it possible to only request assistance from professor/TA, not schedule a meeting.
- Make the design more simple, not showing unnecessary information like team members. Make an own "Team info" button for this.

The first design of the professor page was also made. Figure 5 (a) and (b) shows the student side, while (c) shows the initial design of the Professor/TA page:

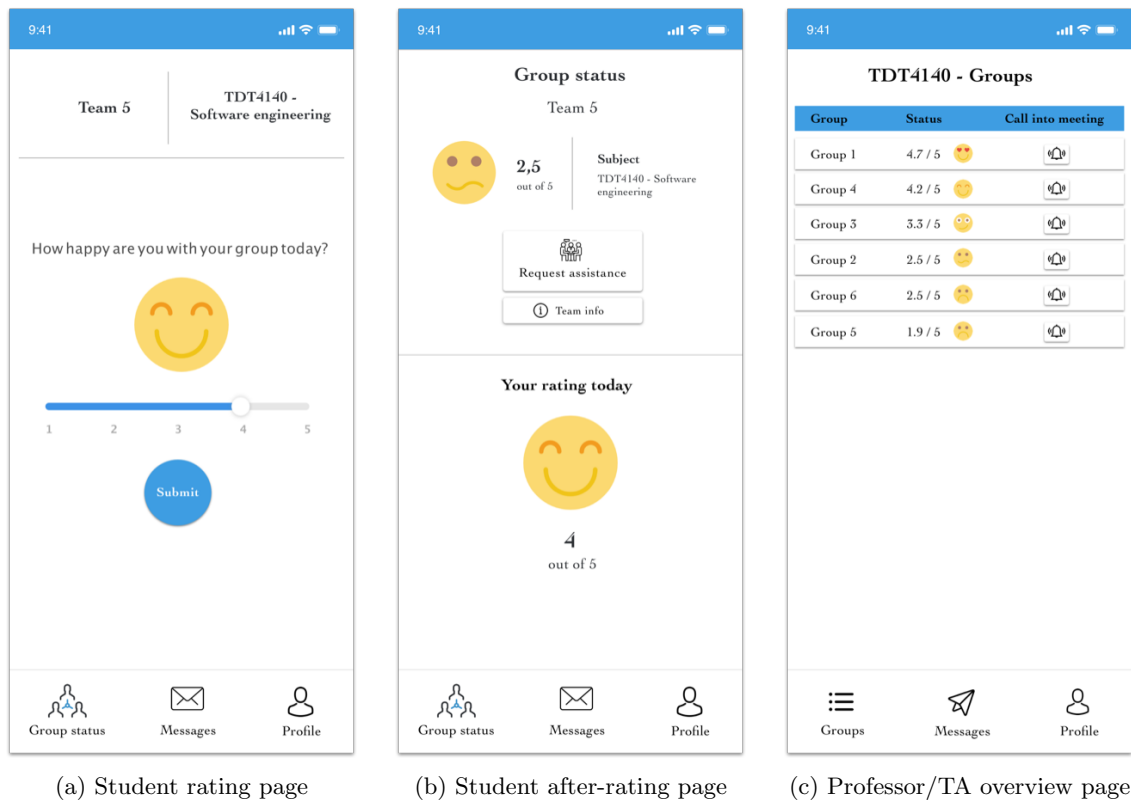


Figure 5: Design iteration 1

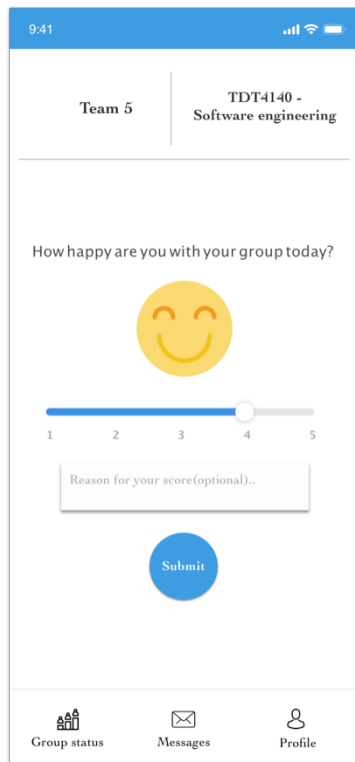
3.2.3 Design iteration 2

After the second design iteration, during the progress meeting and retrospective, guest professor at NTNU Guttorm Sindre was present to provide feedback on the design. The design changes are also based on the results from the student survey.

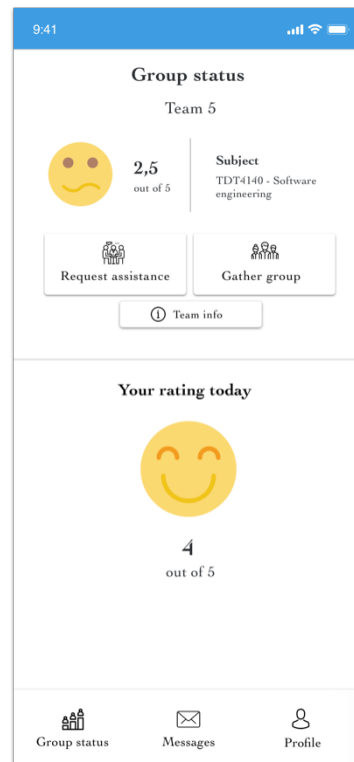
The most important changes from this iteration:

- The groups the instructor/TA are responsible for should be pinned out so they appear at the top of the list. It should also be possible to pin out other teams of interest as well.
- The instructor/TA should have the ability to search for groups and apply filters.
- The instructor should have an overview screen of simple statistics regarding the teams in the course.
- Make a "Gather group" button. This button should make it possible to call your team into a meeting.
- Provide the students with the possibility to say a few words about their score when they rate.
- It should only be necessary and possible for the students to rate the team once a week. This is a change done in regard to the results of the student survey, where the majority answered that they would prefer to rate their team once a week.

The changes are reflected in figure 6 and 7 below:



(a) Student rating page



(b) Student after-rating page

Figure 6: Design iteration 2 - Student side

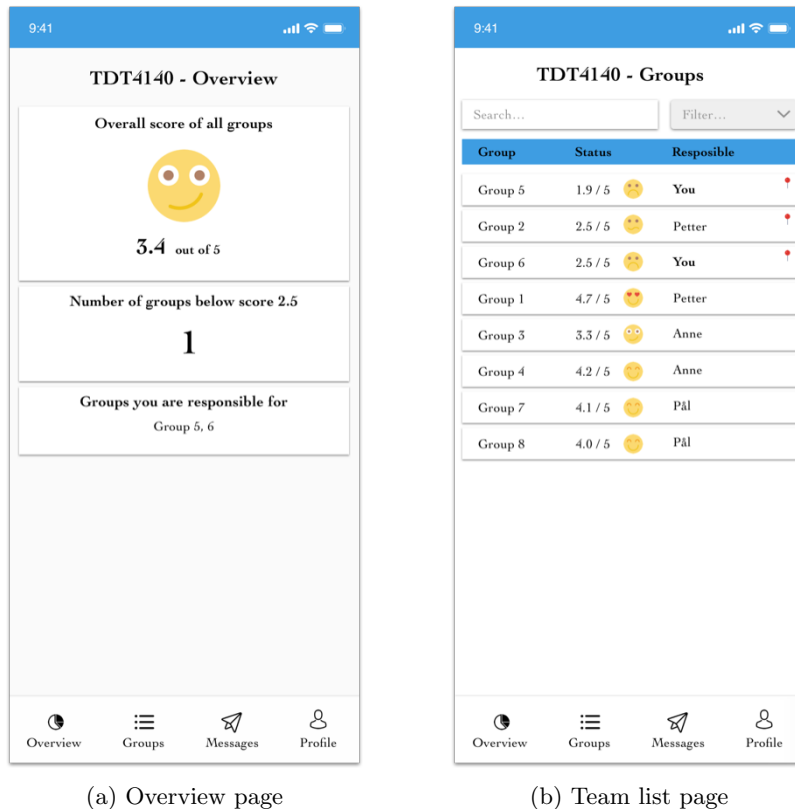


Figure 7: Design iteration 2 - Professor/Teaching assistant side

3.2.4 Design iteration 3

The new changes in design iteration 3 was a result of feedback from interviews with professors and teaching assistants, including the retrospective from the progress meeting. This was the last design iteration before the actual development and programming of the application started.

The most important changes from this iteration:

- The professor page should view the gap between the scores within a team. This means showing the difference between the person that rate the lowest in average vs the person that rated the highest.
- Replace the emoji showing the score with a progress bar in the professor page, and the student status page. Make it possible to click on the different list headers to sort the list based on the header clicked. These changes was a result of the interview with Professor 1, described in the background section.
- The professor/TA should also be able to view individual scores. This change was a direct result of feedback from the teaching assistant interviewed.

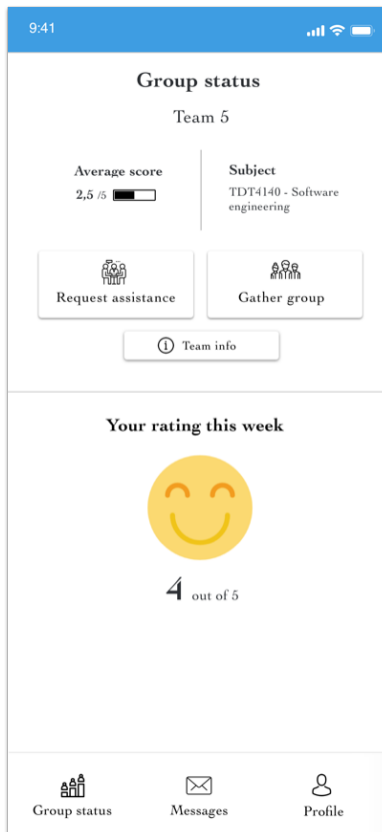
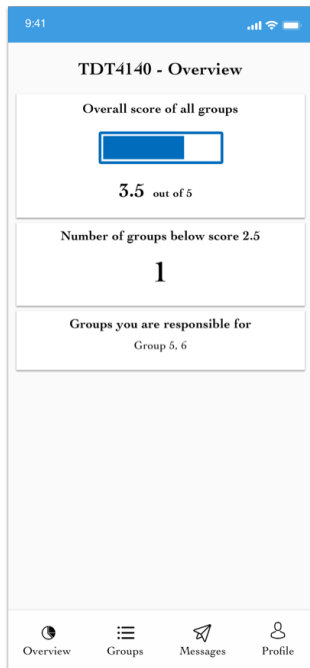
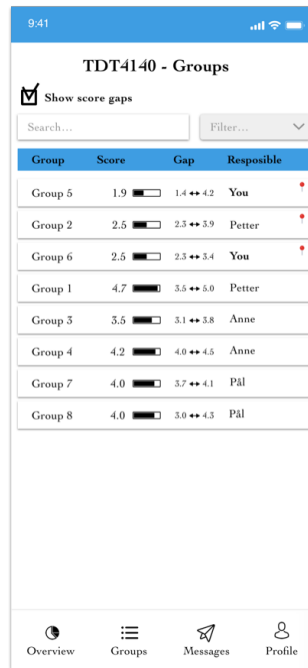


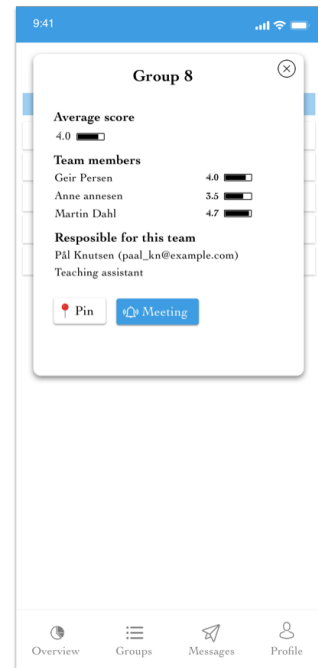
Figure 8: Design iteration 3 - Student after-rating page



(a) Overview page



(b) Team list page



(c) Team info

Figure 9: Design iteration 3 - Professor/Teaching assistant side

4 Architecture

4.1 Overall Architecture

The application is build up in a client - REST API style. The backend and frontend are entirely seperated and independent of each other. One part can be developed without affecting the other. This means increased flexibility and a seamless possibility for connecting other types of clients to the API. The passing of data between the client and server is done using the HTTP protocol. This includes GET requests for reading data, and POST, PUT and DELETE for adding, altering and removing data. The same goes for connecting to the FEIDE API, which is described further under the section "4.6 Feide".

The figure below illustrates the overall architecture:

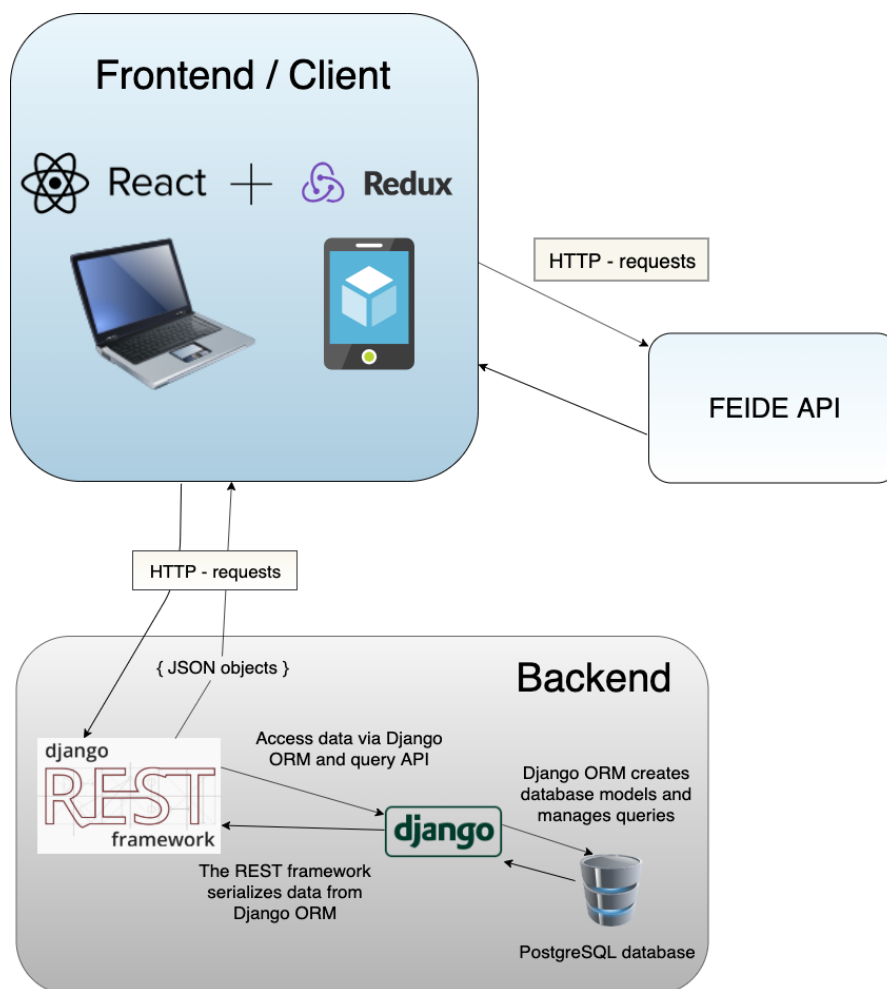


Figure 10: Overall architecture

4.2 Frontend Architecture

The frontend is based on the React-component architecture. Every visible page in the application is a component, with many small components inside including in-component functionality. Redux keeps track of the global state of the application. The files in the project are mainly structured in a way where one component occupies one Javascript file.

The figure on the next page illustrates the overall architecture of the React application. It also illustrates on a very simplified level how the Redux global state is handled. Part "A" marked on the figure shows a simplified illustration of how the react component structure is exploited in the student rating page of the application. The "Navbar", "VerticalContainer", "Button" and "TabBarStudent" tags are self made reusable React-components, and reused on several screens in the application. Part "B" on the figure, illustrates how the NavBar, VerticalContainer and TabBarStudent components are reused in this component as well. In part "C", we also see reusing of some components. To specify the advantage of the reusable-components, for instance, instead of having to create a container that displays it's content vertically-centered on every screen, it is created once and then imported and reused in all screens. In this case the "VerticalContainer" component.

The Redux global state is divided into several parts. Figure 11 below shows the global state of the student attached to the student-side of the application, and the same for the staff. It is worth to notice that every component can attach to every global state and Redux action. This means it would be no problem to attach both the staff and student state to one component.

The global state is accessed through the "props." syntax. In part "C" of figure 11, we can see that "props.teams" is used in the "TeamList" component. This is the data for the teams in one course. We can also see that above this inside the function "getTeamList", "props" is used again, only this time for accessing the Redux-actions to fetch data. Once component "C" renders and shows on the user's screen, the function "props.fetchTeamList()" is called. This is a HTTP GET request that fetches the teams from the Django API. When the response-data is returned from the API to Redux-actions, it is passed to and stored in the Redux global state, specifically in the global state of the staff, under "teams". All components that uses this team data will re-render/reload. Only the team-list will reload, not the entire page, which is mentioned earlier as one of the great advantages with React.

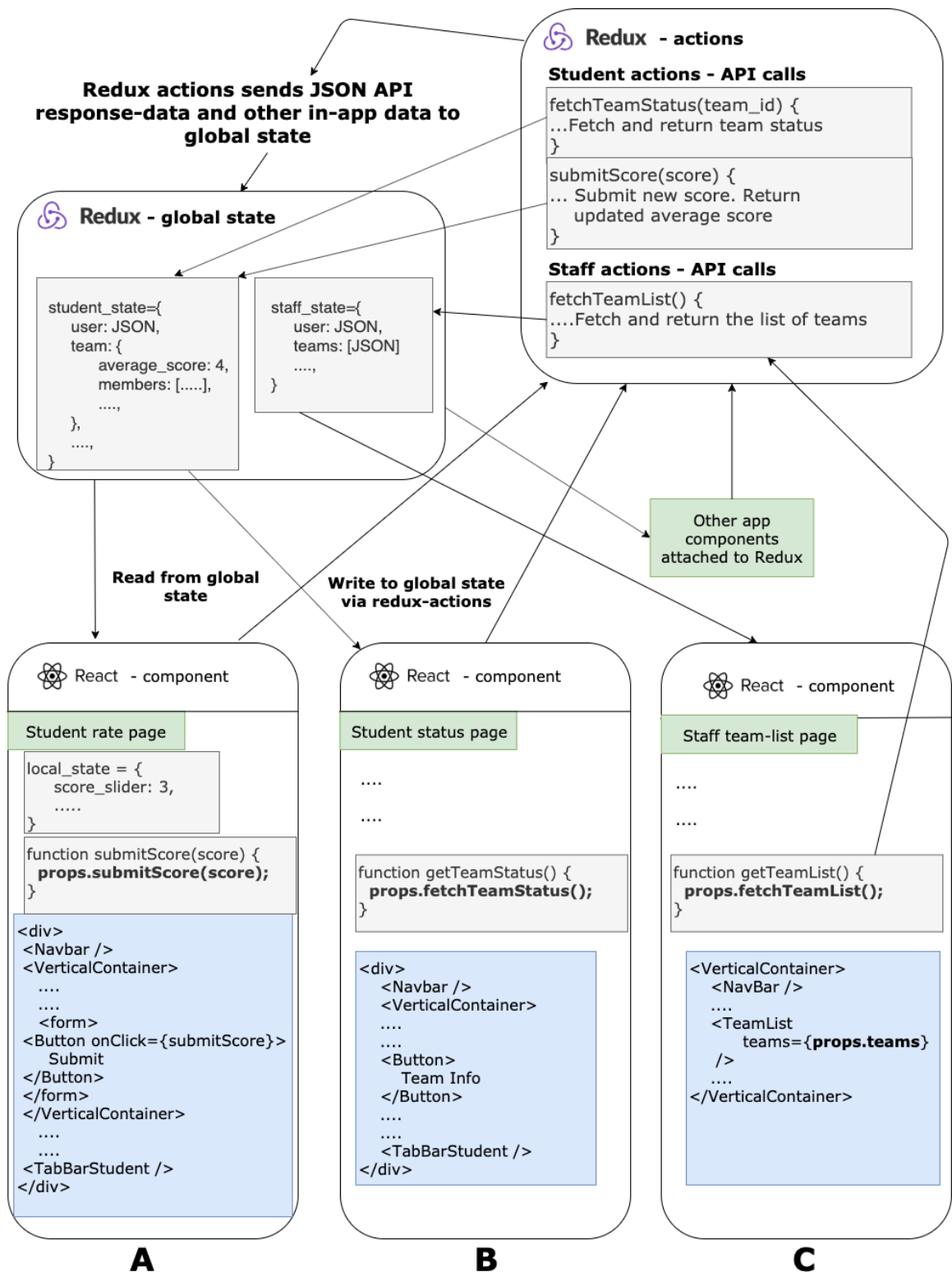


Figure 11: React & Redux architecture - simplified

4.3 Backend Architecture

The backend architecture is built up in REST-ful style. Looking at figure 12 below, it all starts with a request from the client. The request is passed to the API views. Depending on which request is received, it will call that API view. Taking the example from figure 11 with fetching the team list, the request calls the HTTP GET function for the team list. It uses the Django object-relational-mapping-layer (ORM) and Django-queries to get the teams from the database. Once the teams are collected, they are passed to the serializers which converts the teams into JSON. The list of JSON objects are finally returned to the client.

The same process is executed for every API call. Requests that alter the database like POST requests, are in addition to this writing to the database via Django ORM. The updated data is returned to the view and the same process described above follows.

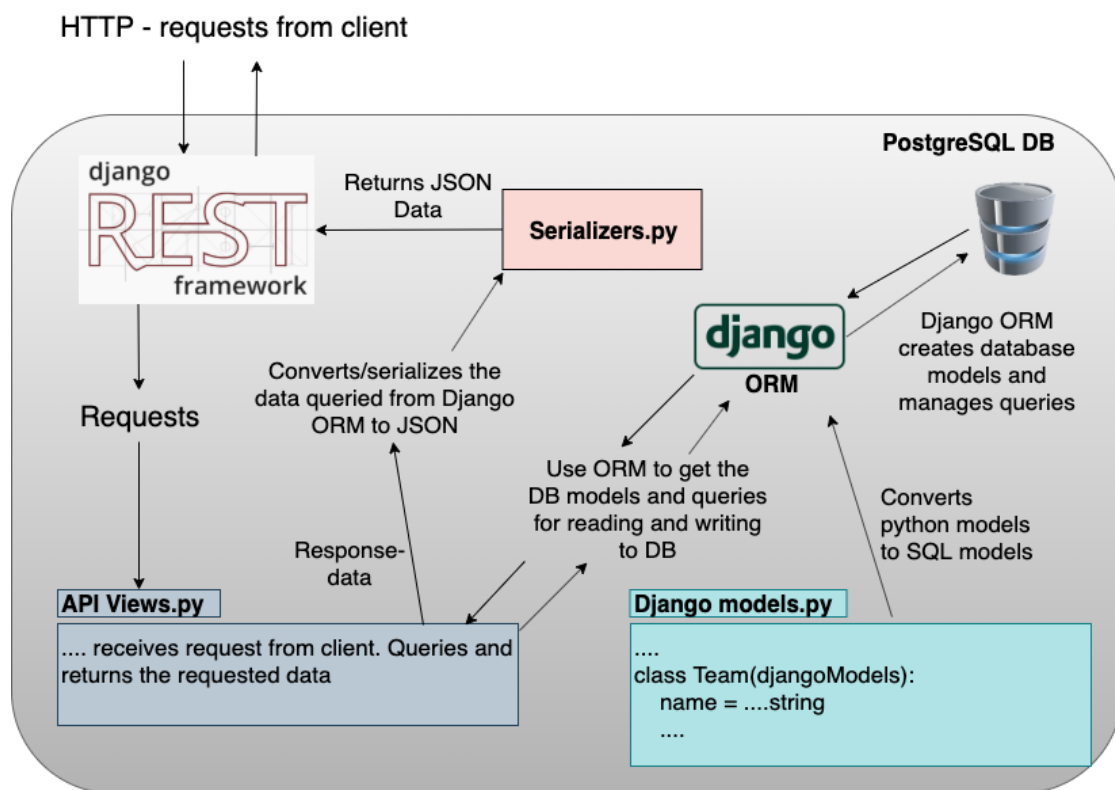


Figure 12: The backend architecture - simplified

4.4 Database Architecture

The database is a relational PostgreSQL database. Figure 13 below shows the internal architecture, the different data models and the relations between them:

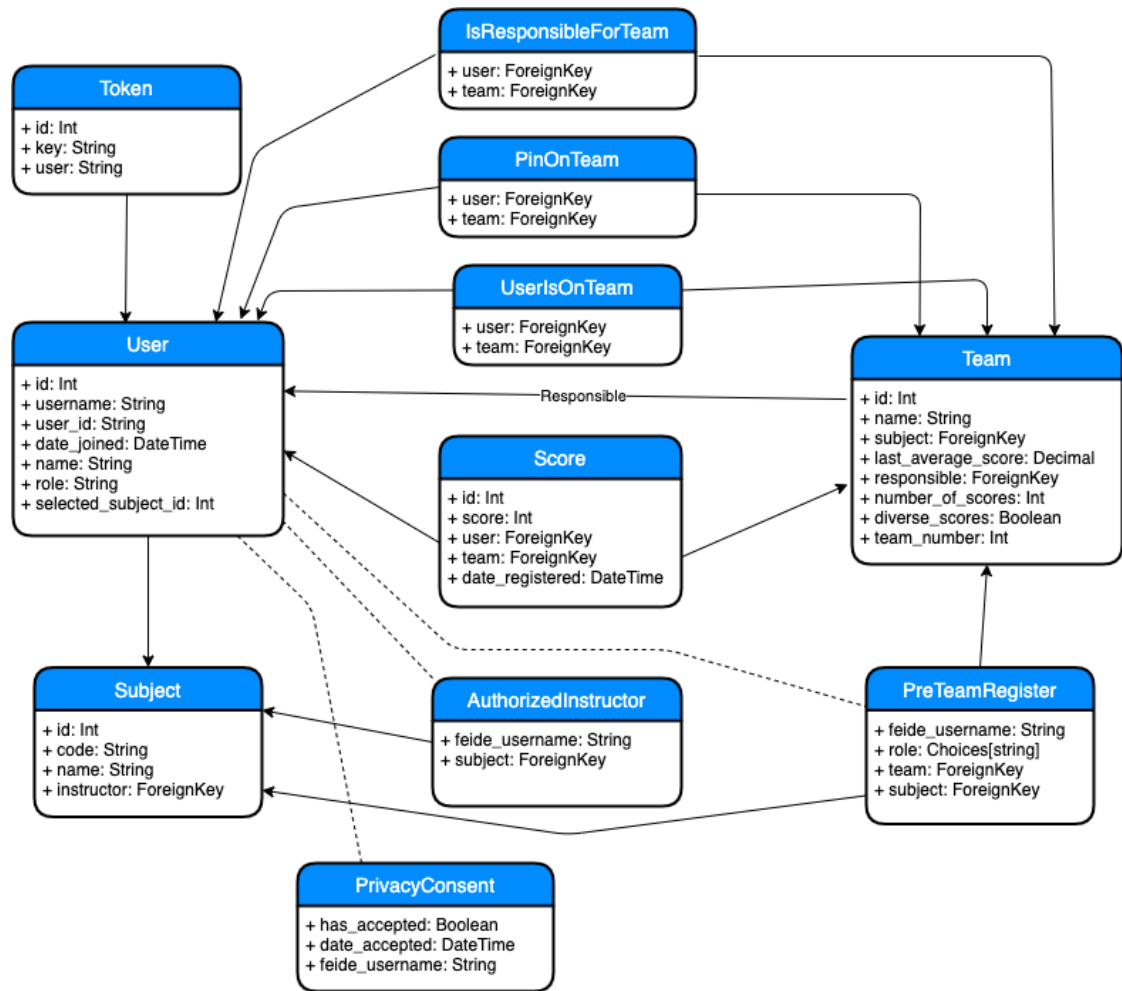


Figure 13: Database architecture

The arrows on the figure indicates which model the foreign keys is pointing to. The dotted lines are not foreign key relations. These are only relations to strings that eventually will exist in a user instance. The reason for this is that these relations need to exist before the user has registered in the database, to have control over which people has access to the relevant course. For instance the "PreTeamRegister". This model is used to register both students and Teaching assistants like an enrollment list. When the user first creates his/her user, the username will be checked with this list, to give the user access to the subject.

4.5 Feide user and authentication

The authentication in the application is handled by using the Feide API. When the user presses the login button in the application, it will redirect the user to the Feide login portal. When the user provides the correct credentials, the Feide API will return an authentication token, which then can be used to fetch information from Feide. This token is then used to fetch the username,

user id and name from Feide. The information along with the token is stored in the application database, and the access token is further used by the application to authenticate requests. Any request that comes into the TeamAccelerator API must have a valid access token to be validated. The access token from FEIDE times out after a couple of hours. When this occurs, the user must log in again with their credentials.

5 Implementation

5.1 Implementation method

Under the implementation phase, much of the same methodology used in the design sprint was utilized. The implementation phase was primarily based on continuous integration. This means that small changes and new features were implemented in small steps, then published directly to the code base ready for re-deployment. The implementation was primarily based on the design sketches, but as the application was tested on various people, and different feedback was taken into consideration, the final product was slightly different from the design sketches.

At the early stages of the implementation, the application was developed in defined sprints. After the first development sprint, a simple version of the student side of the application was developed. The ability to login, choose team and subject was present, and the ability to rate the weekly score and see the team status including a simple profile page. At the first progress meeting it was quickly concluded that a simple first version of the professor side of the application should also be completed to include in the first version. It was also concluded that there should be a simpler way to add information about team and students, instead of having the students to choose team by themselves. This applied to the role selection at the start as well.

After the second implementation iteration a complete working version of the application was done. This included in addition to the parts described in iteration 1, a simple professor side of the application where the professor/TA could view all the teams and their different scores, including individual student scores. In the third implementation iteration, the main focus was to get ready a testable version of the application, so it could be tested on students and teaching assistants. In addition to this, it was needed to make the privacy policy of the app ready, and submit an application to the NSD to get approval of collecting personal data such as name, username and email.

The main focus of the fourth iteration was to get the app published on NTNU servers, handle the privacy policy correctly with NSD, and further improve the app with adding additional features. The rest of the implementation phase was focused on testing the application on students and teaching assistants to get specific feedback and make changes according to this. In addition to this, a lot of time was also spent trying to get approval from NSD of handling personal data, and also moving both the frontend and backend over to a NTNU virtual machine. Both of these processes were time consuming. In general, the latter sprints were less defined than the first ones, and were mainly focused on the same things.

5.2 Choice of technology

5.2.1 Frontend: React & Redux

The frontend part of the application was developed with React and Redux. React is a Javascript library tailor-made for building user interfaces. It is one of the most popular libraries for building frontend applications and is widely used in the IT-industry. The reason for its popularity and the choice of using it in this project is mainly because of the following features:

- React is based on constructing and reusing react-components. A react-component is a self made chunk of code, which can render a view with GUI, and have functionality and state. The key element of react is the ability to break down your interface into small reusable components everywhere in your application which saves you time, gives you structure and less redundant code.
- React is declarative. This means that only the components affected by a data change will re-render, not the entire website which is the case with traditional web-sites. This also makes the code more predictable and easier to debug.
- Once an application is written in React, you can reuse most of the code and turn you application into a native mobile application using React Native.
- Since its popularity, the React-community is large with many contributors, and the documentation is excellent.
- Redux is used to keep control of the global state of the application, instead of keeping all the state inside the components. React and Redux works seamlessly together.

5.2.2 Backend: Django REST Framework

The backend of the application was developed as a REST API with Django and its rest framework. It is a widely used framework, and provides the developer with great possibilities for making web APIs. The main reasons for using Django in this project:

- The rest framework makes serialization of data easy. The serializer classes provided in the framework lets the developer convert data into readable REST-ful formats in a simple way, and make it ready to send to the client. In the case of this project, it is serialized JSON data.
- In Django, you define you database models using python. The object relation mapping layer (ORM) of django converts your python code into SQL.
- Django comes with an own database query API, which is simpler than writing raw SQL queries.

5.2.3 Database: PostgreSQL

The default database provided with Django is SQLite. SQLite does not support concurrency. That means two users writing to the database at the same time. Because of this, the choice landed on using PostgreSQL, which is a powerful and widely used SQL database.

5.3 Code and project structure

View the full source code from the links below:

Backend code repository

https://gitlab.stud.idi.ntnu.no/augustle/master_react

Backend code repository

https://gitlab.stud.idi.ntnu.no/augustle/master_api

5.3.1 Backend

The database models are defined in Python. As described in the architecture section for the backend, the defined python models are converted by the Django ORM to SQL models. Below in figure 14 is a screenshot of the code for the Team model. To convert the python code to SQL, a migration is required, which is simply done by running two separate commands in the terminal.

```
class Team(models.Model):

    name = models.CharField(verbose_name='name', max_length=100)
    subject = models.ForeignKey(Subject, on_delete=models.CASCADE)
    last_average_score = models.DecimalField(max_digits=10, decimal_places=1, default=0)
    responsible = models.ForeignKey(CustomUser, null=True, blank=True, on_delete=models.SET_NULL)
    number_of_scores = models.IntegerField(default=0)
    diverse_scores = models.BooleanField(default=False)
    team_number = models.IntegerField()

    class Meta:
        unique_together = ('name', 'subject',)

    def __str__(self):
        return self.name
```

Figure 14: The Team model

An image from a part of the API endpoint for getting the user model is displayed below:


```

class ApiUser(APIView):

    permission_classes = (permissions.IsAuthenticated, )

    @csrf_exempt
    def get(self, request):
        user = request.user
        serializer = UserSerializer(user, many=False)

        subject = None
        subject_data = None
        if Subject.objects.filter(pk=user.selected_subject_id).count() > 0:
            subject = Subject.objects.get(pk=user.selected_subject_id)
            subject_data = SubjectSerializer(subject, many=False).data

        return_object = {
            'api_user': serializer.data,
            'subject': subject_data
        }

```

Figure 15: Endpoint for getting the user

This endpoint returns the user model in JSON format, as well as the subject model which represents which subject the user currently has selected. The if statement in the code uses the "Subject" class to check if there exists any subject that the user has selected. If this is the case, the subject is queried with "Subject.objects.get(pk=user.selected_subject_id)", where "pk" stands for "primary key", which is the same as id referenced to in figure 13 of the database architecture. This returns the Django ORM model, which needs to be converted to JSON. Further below in figure 15, we can see the "subject_data" variable. Here the "SubjectSerializer" is used to convert the data to JSON format. The same goes for the User class with the "UserSerializer". The boolean field "many=" is just an option to choose whether you want to return a list of JSON objects or a single JSON object. In this case many=False to return a single subject and a single user. The two JSON objects are combined and returned further below in the code which is not visible on the screenshot.

Below is the JSON format of the data returned from the "return_object" from the code above. The UserSerializer and SubjectSerializer wraps the data models into JSON format. As we can see in the figure, the "api_user" field is coming directly from the UserSerializer, and the same applies to the "subject" field with the SubjectSerializer.

```
{
  api_user: {
    pk,
    username,
    date_joined,
    role,
    selected_subject_id,
    name,
    email
  },
  subject: {
    pk,
    code,
    name
  }
}
```

Figure 16: Format of the JSON returned

Below in figure 17 is the serializer for the Subject class.

```
class SubjectSerializer(serializers.ModelSerializer):
    class Meta:
        model = Subject
        fields = ('pk', 'code', 'name',)
```

Figure 17: The SubjectSerializer

Below in figure 18 is the serializer for the User class.

```

class UserSerializer(serializers.ModelSerializer):
    class Meta:
        model = CustomUser
        fields = ('pk', 'username', 'date_joined', 'role', 'selected_subject_id', 'name', 'email')

```

Figure 18: The UserSerializer

All the serializers inherits from "serializers.ModelSerializer". The only thing that needs to be done is specifying in the "Meta" class what fields should be included in the JSON object. If a foreign key field also is needed, this needs to be specified. See the source code for more.

The code below is the HTTP POST request for unselecting a subject from the profile section of the application. This is one of the simplest post requests in the application, but illustrates the simple Django architecture. The user model is accessed from the request object, and can be altered directly with first setting the accessible fields to a desired value, then saving the model. In this example, we can see that the "user.selected_subject_id" is set to "None", as well as the "role" field. Then, these changes are easily applied with the ".save()" function which is a part of the Django ORM and query API. The updated user is then returned from the API to the client in JSON format.

```

class UnselectSubject(APIView):
    @csrf_exempt
    def post(self, request):
        user = request.user
        user.selected_subject_id = None
        user.role = None
        user.save()
        user_data = UserSerializer(user, many=False).data
        return Response(user_data, status=status.HTTP_200_OK)

```

Figure 19: Screenshot of a simple POST request

5.3.2 Backend folder & app structure

The folder structure in the Django project is divided into so called "Django apps". Each app represents a side of the application with associated endpoints. The different apps are: data, staff, student and user. The data app handles most of the database models of the application, the students app handles the student functionality, the staff app handles the professor and TA functionality, and the user is common for both the student and staff app and mostly handles user related functionality. Inside each app folder is the respective views, url's, serializers and models if used, for that side of the application. The "project_api" folder keeps the settings of the project and the main url's. Figure 20 below shows this structure:

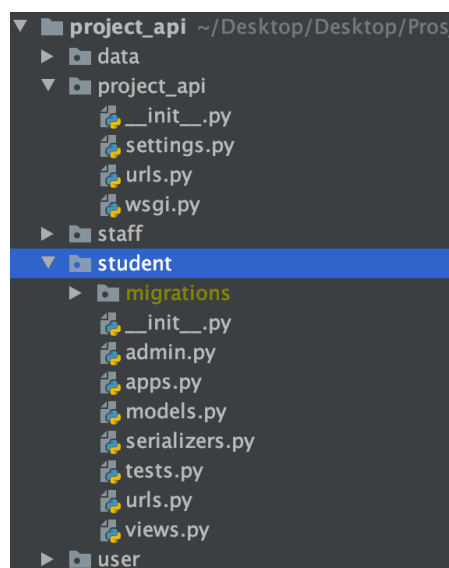


Figure 20: Folder structure of backend project

5.3.3 Frontend folder and & component structure

The frontend code is divided into react components. The most common components are in the "common" folder which is inside the "components" folder. These components are basic components such as buttons and containers. These are not necessarily related to this project and could potentially be used by another react project as well. In the "components" folder there are also project specific components. Every component has its own folder with it's respective Javascript files, CSS styling files and images folder. The different screens/pages in the application are placed in a separate folder "screens". Each screen components represents an app screen and GUI(Graphical User Interface). The "reducer" and "actions" folders are containing Redux functionality, which is overall described under section 4.3 above.

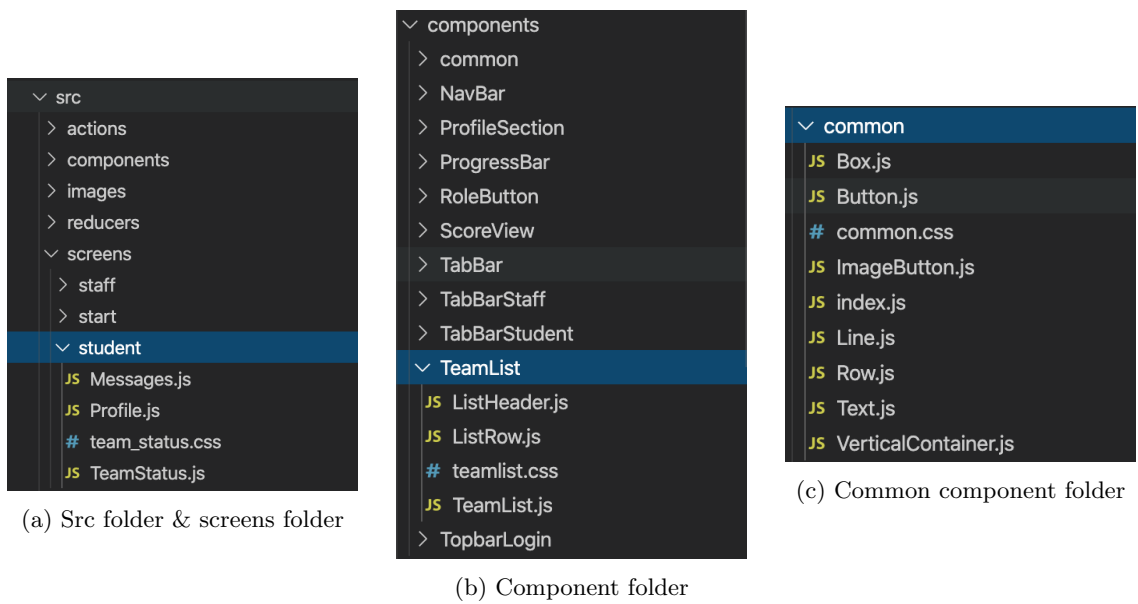


Figure 21: Frontend folder structure

5.3.4 Frontend React code

Below in figure 22 and 23 is the "TeamList" component. This component has a classical react structure used in the application. It contains javascript functionality, JSX code to render and display elements on screen, CSS and some HTML. This component is used to show all the teams and their respective rating score in the staff section of the application. Within this class there is sorting functionality, which is used to sort the list based on the different header parameters. Figure 23 shows how this List component is used and rendered in the code of the staff side of the application.

```

export const TeamList = (props) => {

  const [statusSortVal, setStatusSort] = useState(props.sortVal);
  const [nameSortVal, setNameSort] = useState(null);
  const [respSortVal, setRespSort] = useState(null);

  function scoreSort(a, b) {
    return baseSort(a.last_average_score, b.last_average_score);
  }
  function scoreSortReverse(a, b) {
    return baseSort(b.last_average_score, a.last_average_score);
  }
  function nameSort(a, b) {
    return baseSort(b.team_number, a.team_number);
  }
  function nameSortReverse(a, b) {
    return baseSort(a.team_number, b.team_number);
  }

  function respSort(a, b) {
    return baseSort(b.responsible, a.responsible);
  }
  function respSortReverse(a, b) {
    return baseSort(a.responsible, b.responsible);
  }
}

```

(a) The TeamList component part 1

```

const onClickStatus = () => {
  setNameSort(null);
  setRespSort(null);
  if (statusSortVal >= 2) {
    setStatusSort(null);
  } else {
    setStatusSort(statusSortVal + 1);
  }
}

const onClickName = (b) => {
  setStatusSort(null);
  setRespSort(null);
  if (nameSortVal >= 2) {
    setNameSort(null);
  } else {
    setNameSort(nameSortVal + 1);
  }
}

```

(b) TeamList component part 2

Figure 22: TeamList component

Figure 23: TeamList component

```
const onClickResp = (b) => {
  setStatusSort(null);
  setNameSort(null);
  if (respSortVal >= 2) {
    setRespSort(null);
  } else {
    setRespSort(respSortVal + 1);
  }
}

function pinSort(a, b) {
  return boolSort(b.pinned, a.pinned);
}

const getSortedList = (sortVal, sortFunction, sortFunctionRev) => {
  let teams = [].concat(props.teams);
  teams = teams.sort(pinSort);
  if (sortVal === 1) {
    teams = teams.sort(sortFunction);
  } if (sortVal === 2) {
    teams = teams.sort(sortFunctionRev);
  }
  return teams;
}
```

(a) TeamList component part 3

```
const List = () => {
  let teams = [].concat(props.teams);
  if (statusSortVal > 0)
    teams = getSortedList(statusSortVal, scoreSort, scoreSortReverse);
  if (nameSortVal > 0)
    teams = getSortedList(nameSortVal, nameSort, nameSortReverse);
  if (respSortVal > 0)
    teams = getSortedList(respSortVal, respSort, respSortReverse);
  const team_list = teams.map((team) => (
    <ListRow onClick={() => props.onClick(team.pk)} key={team.pk} team={team} />
  ));
  return (
    <div className='listContainer' style={props.style}>
      <ListHeader
        statusSortVal={statusSortVal}
        respSortVal={respSortVal}
        nameSortVal={nameSortVal}
        onClickStatus={() => onClickStatus(statusSortVal)}
        onClickName={() => onClickName(nameSortVal)}
        onClickResp={() => onClickResp(respSortVal)}
      />
      {team_list}
    </div>
  );
}

return (
  <VerticalContainer style={{ width: '100%' }}>
    <List />
  </VerticalContainer>
);
}
```

(b) TeamList component part 4

```

return (
  <VerticalContainer>
    <NavBar />
    {props.subject && (
      <Text bold size='22px' style={{ margin: '15px' }}>{props.subject.code} - Overview</Text>
    )}
    {(!props.loading_fetch && props.staff_team_list) ? (
      <TeamList onClick={(team_id) => onTeamClick(team_id)} teams={props.staff_team_list} />
    ) : (
      <Loader />
    )}
  )}
)

```

Figure 24: Usage of TeamList component

As seen in the code of the "TeamList" component, this component also uses other smaller components like "ListRow" and "ListHeader". Figure 24 shows the easy usage of the TeamList component. It can be reused anywhere in the application with one line of code.

5.4 Hosting the application for testing with Heroku and Surge

Before the application was ready for production and testing on an actual NTNU course, there was need for testing the application qualitative directly on people. For this matter, there was no need to have real users and data, because it was the mere design and functionality of the application that was going to be tested. The way this was done was to deploy a testversion of the application on the free services Heroku and Surge.

5.4.1 Heroku for backend

Heroku is a cloud platform as a service(PaaS). It is used by many companies both in production and for testing, and it supports several programming languages including Python and Django. This simple service lets the developer deploy an application to the cloud very effectively and without any domain. The domain is generated by Heroku. This service also lets the developer easily integrate a PostgreSQL database, which is used in the TeamAccelerator application.

5.4.2 Surge for frontend

Surge is a service for publishing frontend applications, just within seconds. This is especially optimal for React applications. The only actions the developer has to do to get an application published to the internet is one simple command in the terminal. Surge gives you a URL and the application is up and running. This service was optimal for publishing the application for the purpose of testing.

5.5 Hosting the application for production

Publishing the application for production was a substantially more comprehensive task. Both the frontend and backend application was to be hosted on a NTNU virtual machine. This machine is a Ubuntu 18.04 machine. There were certain permissions that was needed to be bypassed in order to get the application up and running on the Ubuntu machine with the specified domains. The backend application is running on the Ubuntu machine with Nginx and Gunicorn. Gunicorn is a Python Web server gateway interface HTTP server. In shorter terms, Gunicorn is the HTTP server for the application, and serves the Django API. Nginx is a high performance HTTP server

and reverse proxy. In the case of this application, Nginx is used as a reversed proxy to Gunicorn, giving the application access to the security features and performance of Nginx. A reverse proxy means that Nginx sits in front of Gunicorn, and forwards the client requests to Gunicorn.

5.6 Handling the course permissions

A challenge with testing the application on an actual course was to control which people had access to the given course. Only the people enrolled in the course and the people responsible for the course should have access to the course. The optimal solution to handle this would be to automatically fetch this data from Blackboard, FEIDE or another data source that sits on this information. Unluckily, this turned out to be too comprehensive and not possible to achieve within the time frame of the spring.

5.6.1 First solution

The first tried access solution of the app was letting the user select their role, the subject and which team they belong to. In the Appendix part A below, this login process is illustrated. To avoid having people not enrolled in a course registering in a course, a temporary password solution was implemented. Every team has a password, and it is only possible to register to a team if the user has the correct password.

It was quickly concluded that this login process was consisting of too many steps, and would probably prevent the user from taking the effort to register in the application. In addition to this, there was no access control to the staff side of the application. This means that every FEIDE user could potentially login as a teaching assistant or professor, and view all the course data.

5.6.2 Second solution

In the second solution of the access control, role selection was controlled and there were no password protection on the teams. This means that only users that are pre-registered in the database with staff access, can register as a teaching assistant or a professor. As there are usually not a substantially large amount of professors and teaching assistants, these people could easily be registered manually in the database. The password protection of the teams was removed. The reason for this was the potential challenge in handling and distributing passwords to all teams in a course. Something more had to be done with the access control. This solution would let anybody register to a team in a course.

5.6.3 Current solution

The current solution handles the access control of the app in a way that only lets the people with the right permissions register in the application. The way this works, is with a pre-defined list of all students in a course, and which team they belong to. With instructor permissions, all this information is possible to download from blackboard. The information is converted over to a simple .txt file, that can be uploaded in the application. The format of the data is simply the team name, followed by a comma and the FEIDE username, separated line by line for each student. This is illustrated below, where each item represents a line in the txt file:

- Team_1, user1
- Team_2, user2.

- Team_2, user3
- Team_1, user4
- ..

A user with instructor permissions has the ability to upload a txt file in this format. The application converts this to JSON data, and sends it to the backend. This information is then registered in the "PreTeamRegister" model. If a given team in a row of this txt file does not exist, it will be created. In figure 25 below, this process is illustrated. It starts with selecting and uploading the desired .txt file in the format described above. When the user hits the "Submit teamlist" button, the text in the txt file will be converted to JSON format, as illustrated in the figure. The backend will receive this HTTPS POST request, with the entire list of JSON objects. These objects are then iterated through, creating all the team objects in the list as well as the "PreTeamRegisters" for all the students in the list.

When a student then registers in the application, the student will be directly enrolled and registered in the course/subject and the team. The application validates the FEIDE username of the newly registered user with all the PreTeamRegister objects, and assigns the user to the correct team. The user has to select the subject from a list. A user could potentially be enrolled in many subjects with group work projects. Staff permissions for TA's and professors must be manually registered in the admin backend panel of the application. Since there are not so many of them in each course, this can be done fairly quickly.

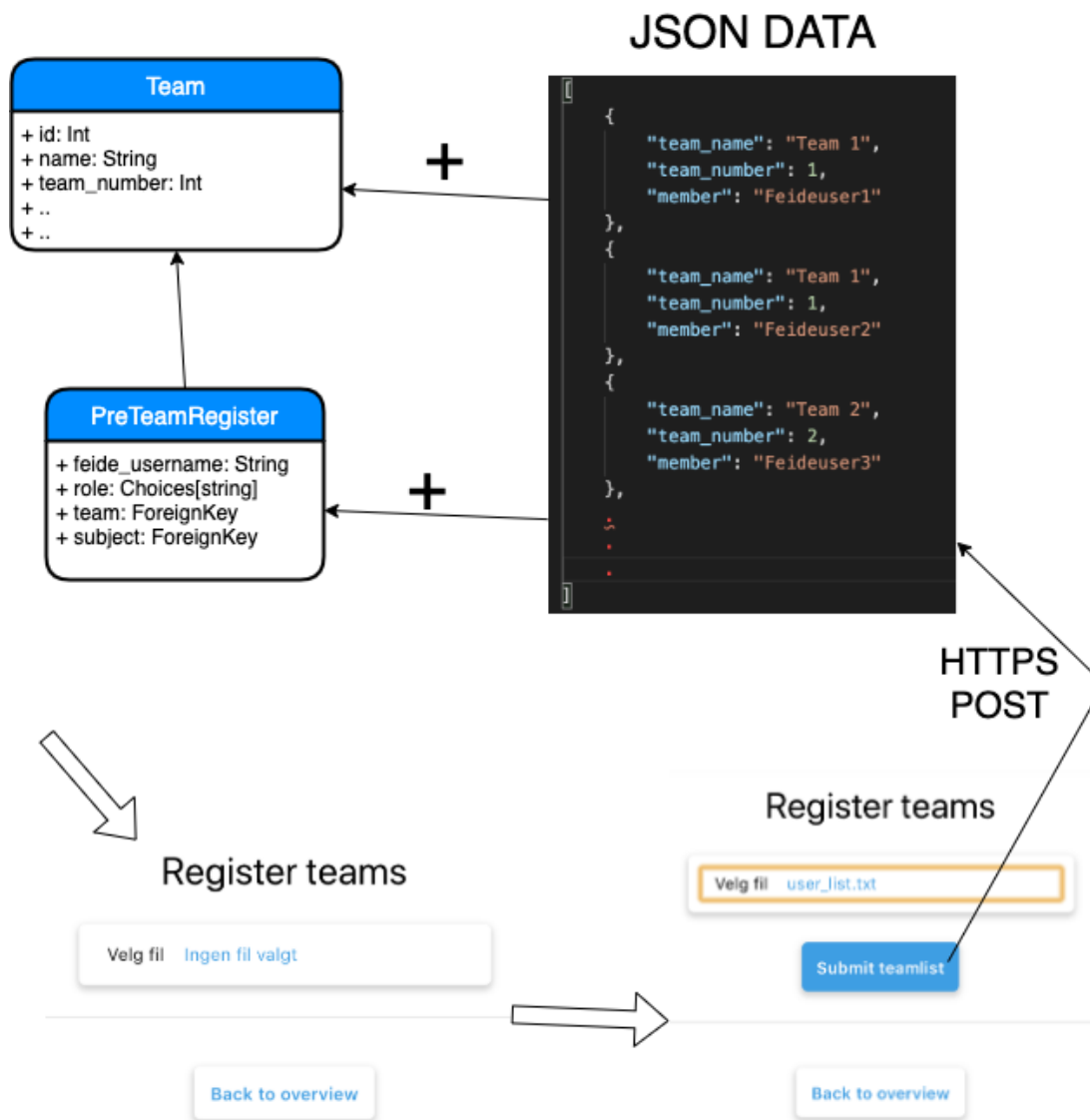


Figure 25: Access control upload flow

6 Privacy Policy and GDPR

6.1 GDPR Compliance

To comply with the privacy policy and the requirements of the EU GDPR rules, specific functionality was implemented. According to the GDPR, the user should have the ability to delete themselves and all related data at any point in time. This was implemented in the application under the profile section, and is illustrated in figure 26 below:

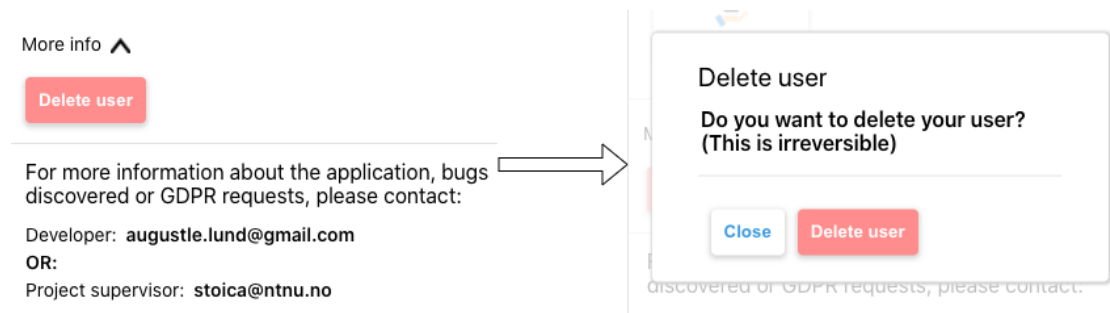


Figure 26: Delete user in app

According to GDPR, the user should also have the ability to ask for all the data that is stored in the database, and the ability to export it on a readable format. As seen in figure 26, the user has the ability to make email contact in order to request this. As the application in this scenario do not have a large amount of users, this way of handling GDPR requests is sufficient. On the other hand, if the application were to be expanded into several courses with a large amount of users, it might would have been necessary to implement a page in the application where the users had the ability to view all data and export it.

The data transferred from client to backend and back is SSL encrypted.

6.2 NSD

In order to be allowed to collect user data through the application for research purposes on NTNU, an application had to be approved by the Norwegian Centre for Research Data. An application was sent to the NSD. The processing and approval time of this application was about 3-4 weeks. After this, the app could be deployed to the students and the staff at the NTNU course TDT4180 - Human Computer Interaction.

6.3 Privacy Policy In-App

When the user logs in for the first time through FEIDE, the user will immediately be asked if he/she agrees to the overall privacy policy of the application. This means agreeing to let the application collect their FEIDE username, email and full name. In figure 27 below, is a screenshot of this. The user will not have the ability to create a profile if they choose not to accept the policy. The "Create profile" button will only appear when the checkbox is checked.

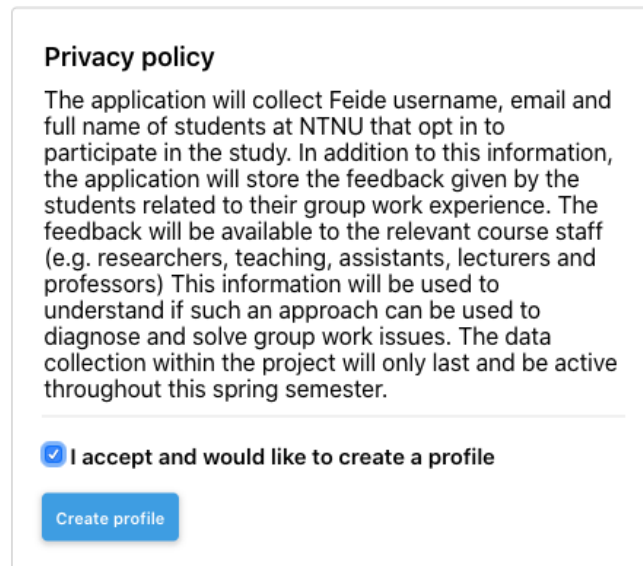


Figure 27: Privacy policy in app

7 Results and future work

7.1 The application

The final product is a fully functional application for both students and course staff. Course staff includes teaching assistants, professors and other assistants that have some kind of role in managing the course. The application will be described below with screenshots. The data in the screenshot is only test data, but will give a sufficient description and illustration of the application.

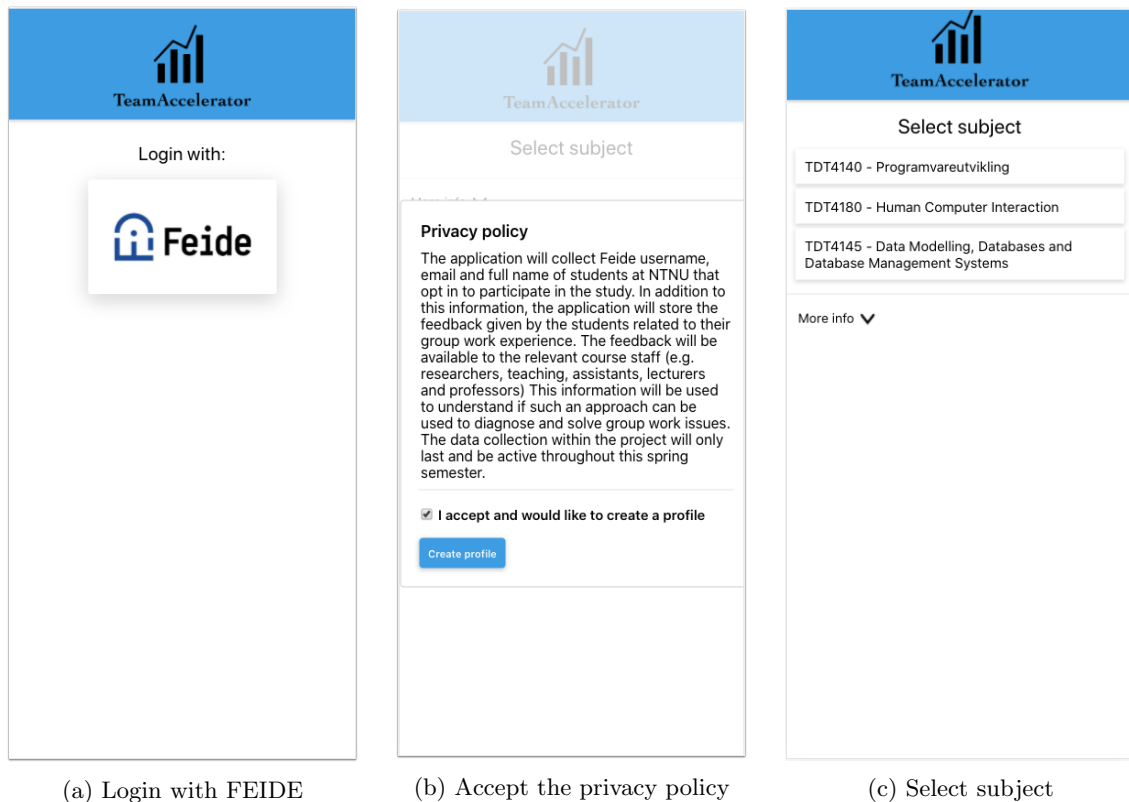


Figure 28: Login process

7.1.1 The registration process

- (a) - The first page that meets the user is the Feide login button. When the user clicks this button, the user will be redirected to the FEIDE login portal. When the users type in their credentials and hit enter, they will be redirected directly to the privacy policy screen in the figure

- (b) - Unless the user has checked the checkbox that he/she is willing to let TeamAccelerator store data about them, it is not possible to go any further. The privacy policy has to be accepted.
- (c) - Once the privacy policy is accepted, the users have the ability to select which course they want enroll in.

7.1.2 Student in-app functionality

Below are the different features of the student side of the application.

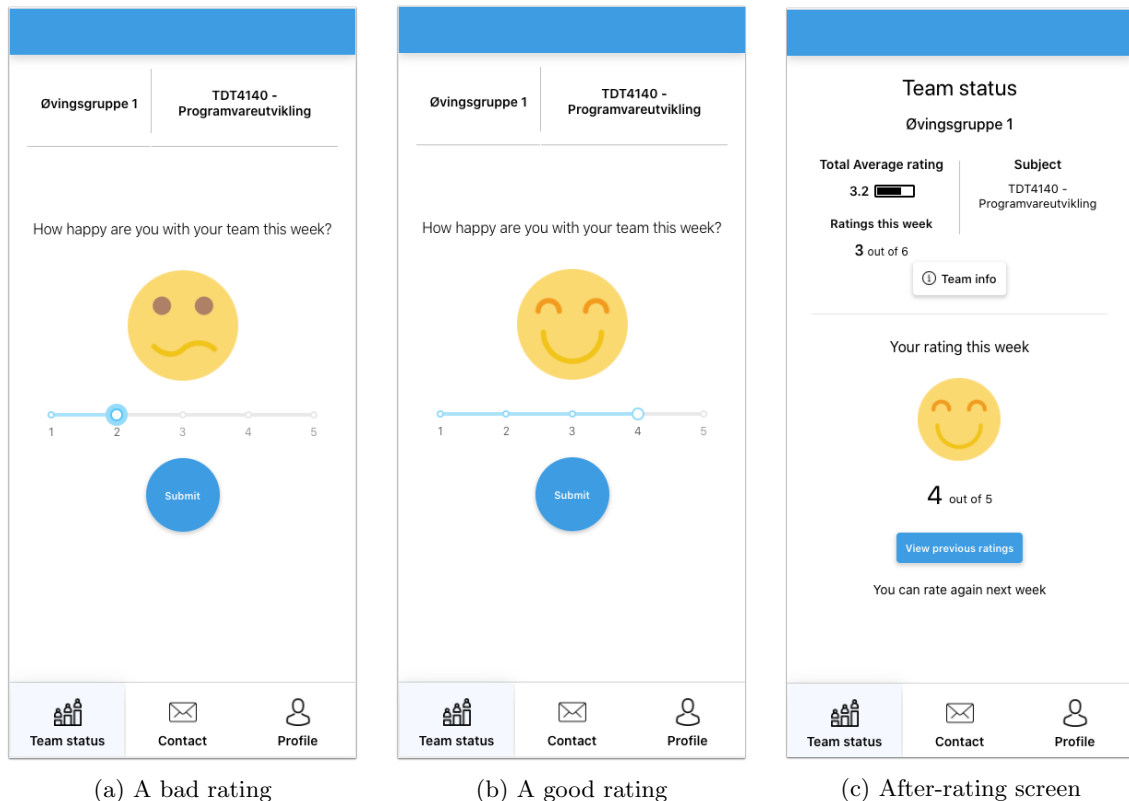
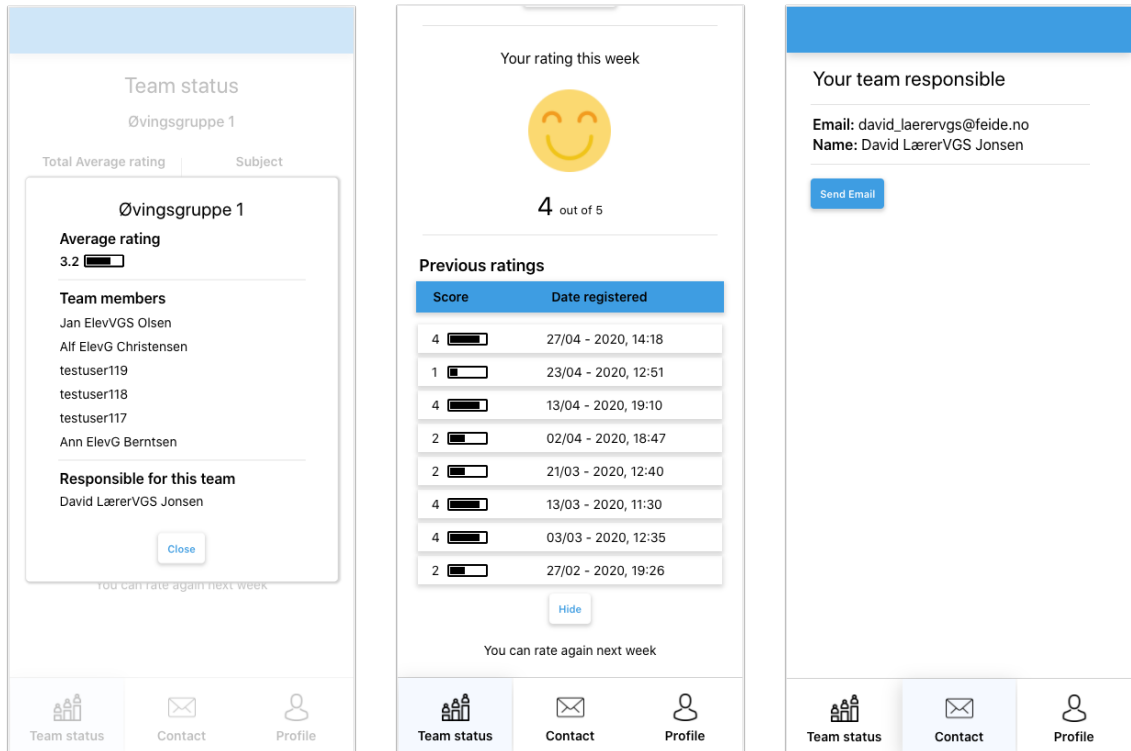


Figure 29: The rating side for students

- The ability to register the weekly score to your team. It is only possible to rate once a week.
- (a) - If the student is not pleased with his/her team, the natural thing to do would be to give a bad rating. In this case that is 2 out of 5.
- (b) - The same applies to a student that is pleased. In this case, 4 out of 5 is a good score.
- (c) - After rating, the team overview page appears. Here the user can view the basic information of the team, and see the average score and the last score of the user registered. The user can also see how many people on the team that has rated the current week. In this case there are 3 team members that has rated.

- Basic information about which course is selected, the total average rating of the team and team name is displayed at the top.



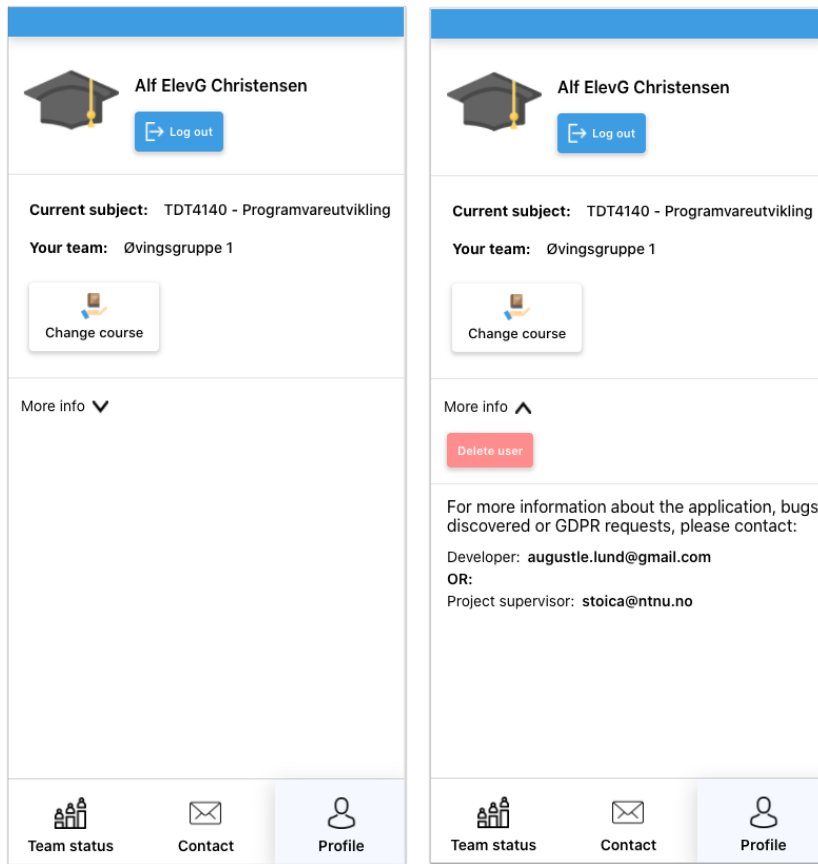
(a) Show team information

(b) See your previous ratings

(c) Contact the teaching assistant

Figure 30: Student side

- (a) - The student can view information about the team members, the total average score, and the responsible teaching assistant for the team, with full name.
- (b) - The students can view their previous ratings, and the date they were registered.
- (c) - The app provides a simple gateway to contact the teaching assistants. When the student click on the "Send Email" button, the native email application will automatically open on the platform the students are using, with the teaching assistant as the recipient.



(a) Hiding extra information by default (b) Extra information expanded

Figure 31: Student profile page

- In the profile page, the basic information about the selected course and which team the student belongs to is displayed.
- The user logs out of the application by clicking the "Log out" button at the top.
- The "More info" section is by default hidden. When clicked, it expands to view more information. The user has the ability to delete themselves entirely. This process is described a bit more in detail in section 6.1.

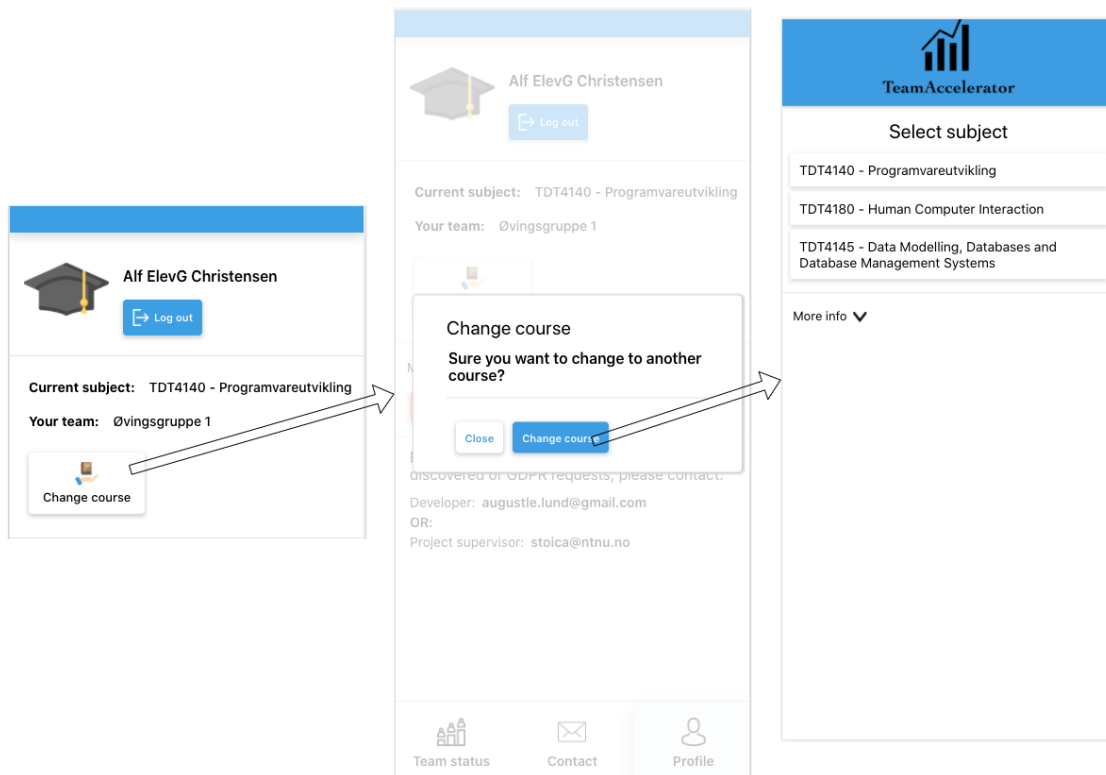


Figure 32: Changing to another course

- In the potential case where a student is enrolled in several project based courses, the students can easily change to another course by clicking the "Change course" button.
- It is not possible to select a course which the student is not pre-registered in. This process is described in section 5.5.3.

7.1.3 Staff in-app functionality

Described below is the functionality of the staff side of the application, with images.

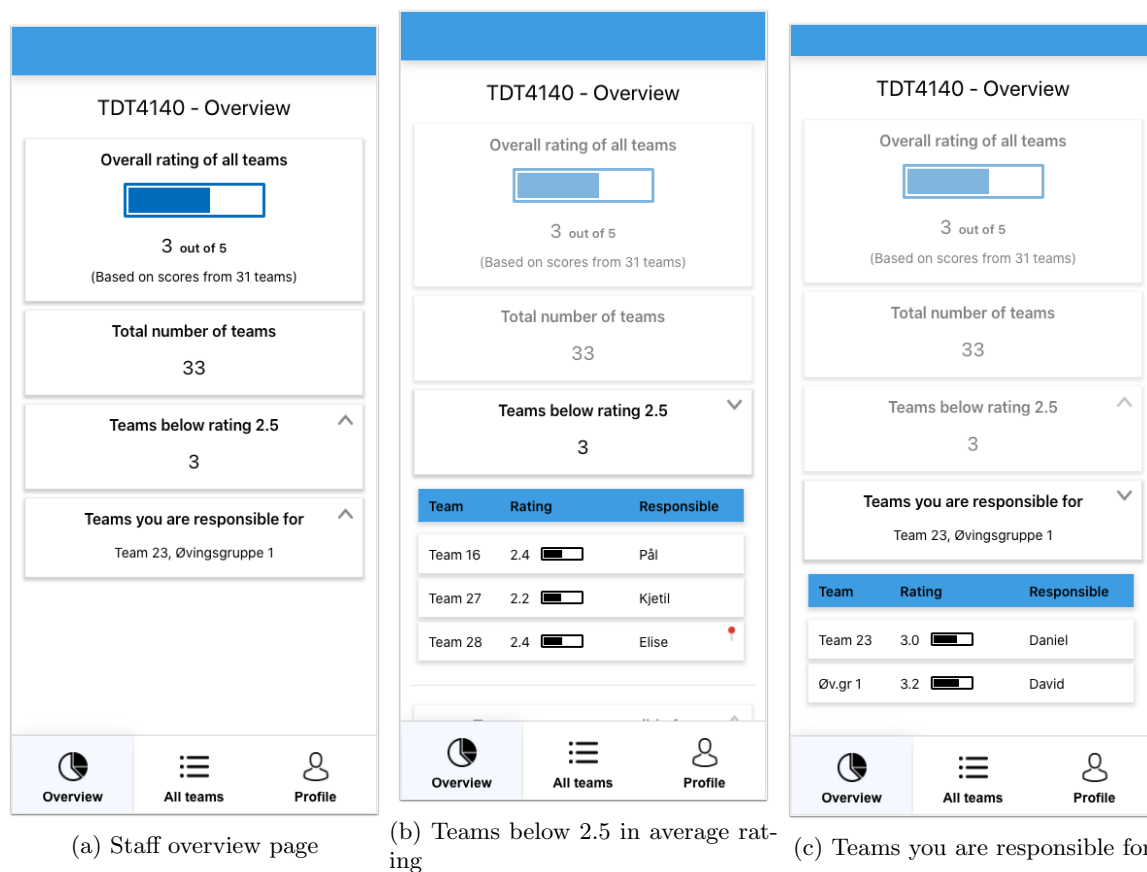
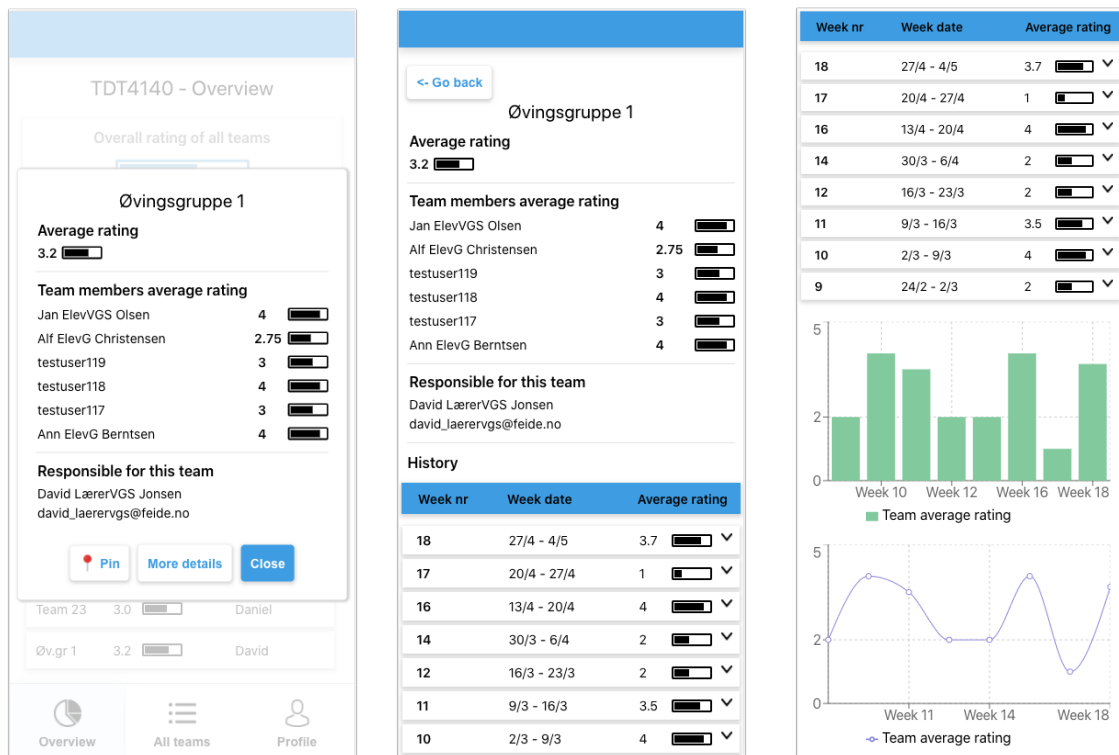


Figure 33: Staff side of the application

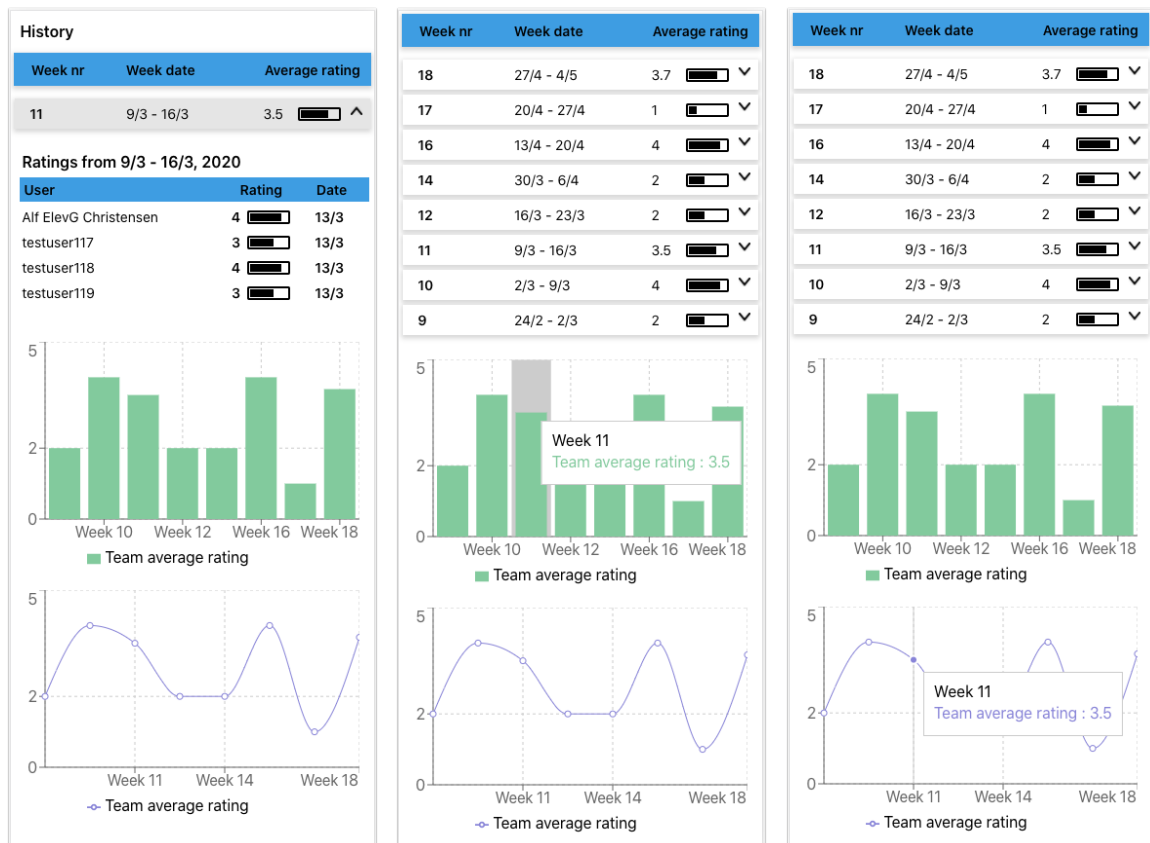
- (a) - The ability to view an overview of basic statistics of all the teams in the selected subject. These statistics are the average score of all teams, total number of teams, number of teams below score 2.5 and what teams you are responsible for.
- (b) - In the overview section, the user can click on the box to view the teams below 2.5 in rating.
- (c) - The user can also click on the box of the responsible teams to view the specific teams



(a) Click on a team (b) View more details of the team (c) Simple graph representation

Figure 34: Continuation of staff side of the application

- (a) - The user can click on a team appearing in the dropdown list when it is expanded in the overview section. This will make more information about the team appear on the screen, as well as some actions.
- (b) - When the user clicks on the "More Details" button, he/she will be redirected to a new page with more information about the team.
- (c) - The detail page shows a simple graph representation of the history of the rating of the team. First a bar chart, then a line chart. These graphs represents the same numbers. They show the average rating of the team, every week someone has rated the team. The list above the graphs also shows some more information. It is possible to click on one week to view even more information, which will be illustrated below.



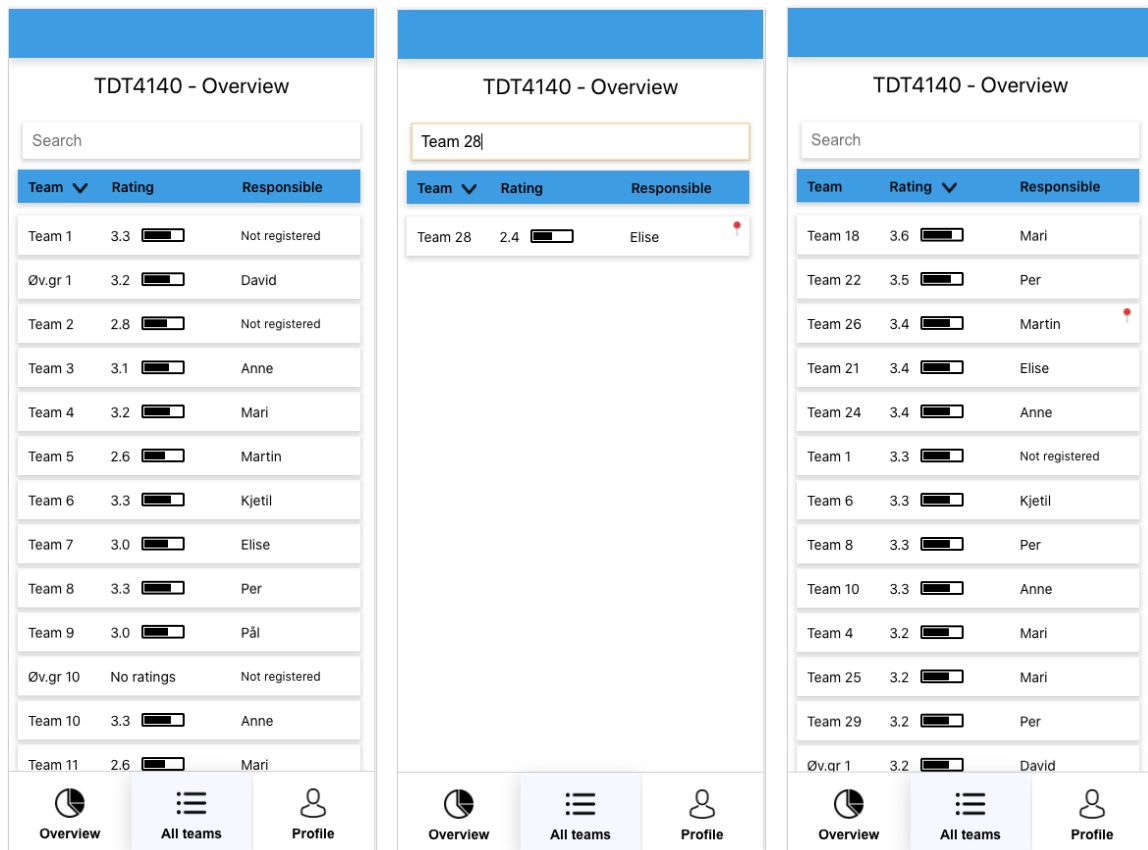
(a) View a specific week

(b) Click on a bar

(c) Click on data point

Figure 35: Continuation of staff side of the application

- (a) - The user can click on a specific week to view all the individual ratings of the team members that particular week, and the date.
- (b) & (c) - The user can highlight the bars and click on the data points on the line chart to see the details.



(a) View a list of all teams

(b) Search for a team

(c) Sort the list

Figure 36: Continuation of staff side of the application

- (a) - View a list of all the teams in the course, the average rating of the team, and the responsible teaching assistant for the team.
- (c) - It is possible to sort the list based on all the headers. In this case the list is sorted by rating in a descending order. It can also be sorted in a ascending order, by clicking the header once more. As seen in the (a) figure, the list is sorted by team number, ascending.

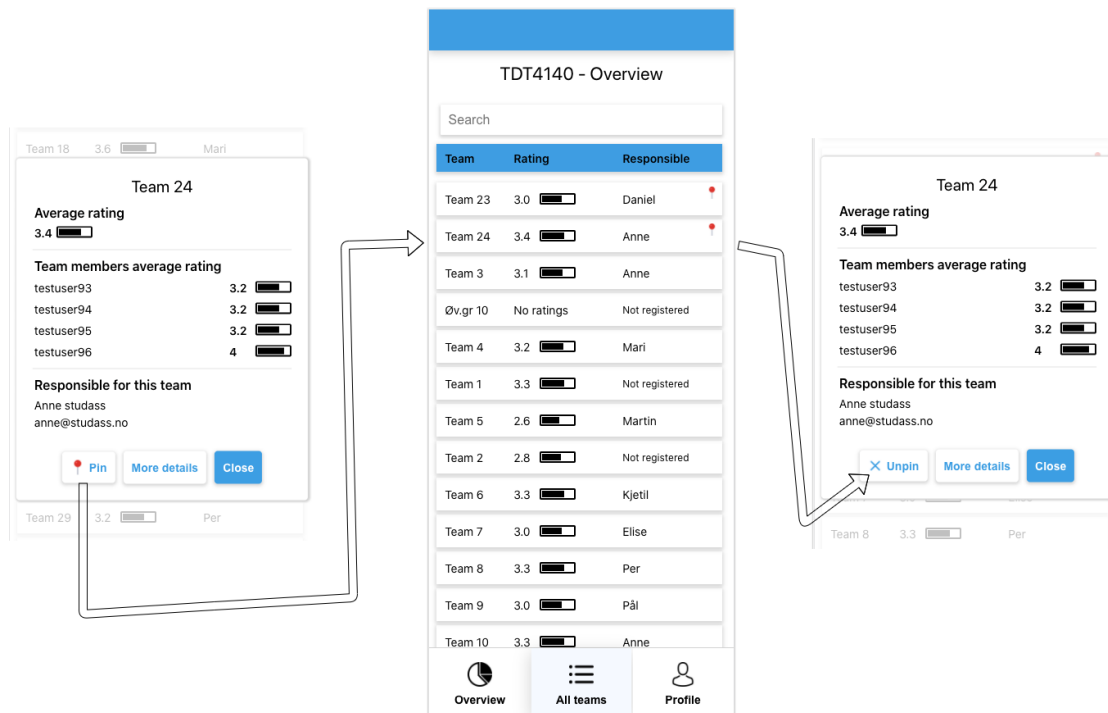


Figure 37: Pin and unpin teams

- To gain a better overview of the teams, it is possible to pin a team, so that it appears at the top of the list. In the same way, it is possible to unpin a pinned team, by clicking the team and then "unpin". All the teams the user are responsible for will always appear at the top of the list as pinned.

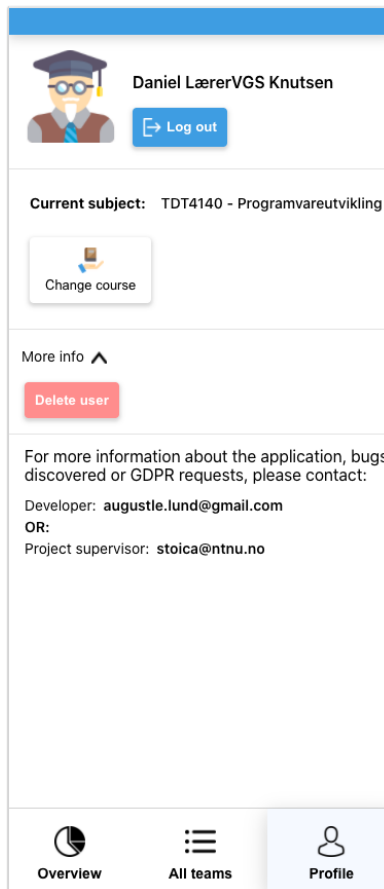


Figure 38: Staff profile

The profile page of the staff side of the application is the exact same as the student profile side, illustrated in figure 31 and 32. The only difference is that there is naturally no team to display in the staff profile, and the profile logo is different.

7.2 Django backend admin panel

Django provides the developer with a backend panel that allows the developer to manage the database and access all the data. It also allows the developer to customize the site with simple Python code. The admin panel shows all the data models in the database, and it is possible to click into each instance and do changes if necessary.

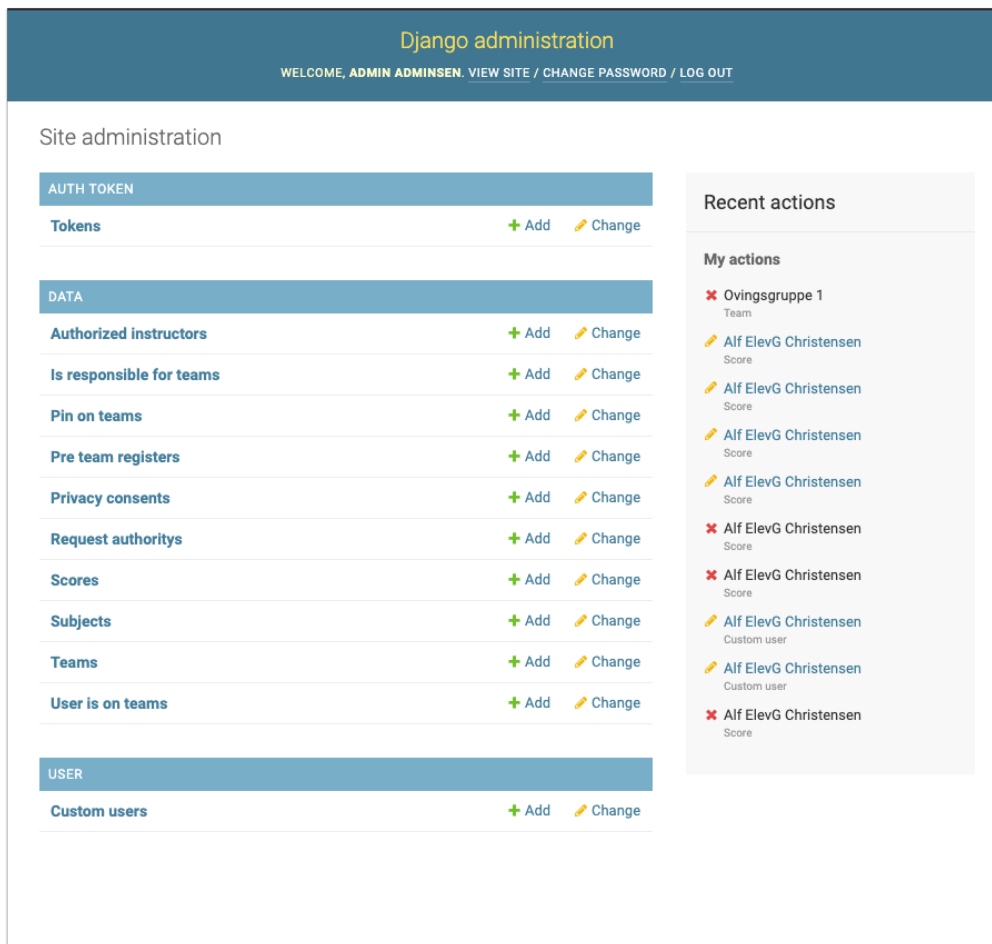


Figure 39: Admin panel, main overview

The overview in figure 39 gives the user control over all the data models in the database.

Django administration WELCOME, ADMIN ADMINSEN. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home > User > Custom users

✔ Successfully deleted 1 custom user.

Select custom user to change ADD CUSTOM USER +

Q Search

Action: Go 0 of 100 selected

<input type="checkbox"/>	USERNAME	DATE_JOINED	IS ADMIN	NAME	ROLE
<input type="checkbox"/>	Admin	March 2, 2020, 1:45 p.m.	✔	Admin Adminsen	-
<input type="checkbox"/>	alf_elevg	March 8, 2020, 6:25 p.m.	✘	Alf ElevG Christensen	-
<input type="checkbox"/>	ann_elevg	April 27, 2020, 12:12 p.m.	✘	Ann ElevG Berntsen	Student
<input type="checkbox"/>	studass3	March 2, 2020, 2:03 p.m.	✘	Anne studass	Teaching assistant
<input type="checkbox"/>	asbjorn_elevg	March 2, 2020, 2:21 p.m.	✘	Asbjørn ElevG Hansen	Student
<input type="checkbox"/>	daniel_laerervgs	March 8, 2020, 6:27 p.m.	✘	Daniel LærerVGS Knutsen	Teaching assistant
<input type="checkbox"/>	david_laerervgs	March 8, 2020, 6:44 p.m.	✘	David LærerVGS Jonsen	Teaching assistant
<input type="checkbox"/>	studass7	March 2, 2020, 2:07 p.m.	✘	Elise studass	Teaching assistant
<input type="checkbox"/>	eva_student	March 3, 2020, 5:21 p.m.	✘	Eva Student Åsen	Student
<input type="checkbox"/>	frank_foreleser	March 2, 2020, 4:51 p.m.	✘	Frank Foreleser Føllesen	Instructor
<input type="checkbox"/>	jan_elevvgs	March 3, 2020, 5:27 p.m.	✘	Jan ElevVGS Olsen	Student
<input type="checkbox"/>	studass6	March 2, 2020, 2:05 p.m.	✘	Kjetil studass	Teaching assistant
<input type="checkbox"/>	studass4	March 2, 2020, 2:04 p.m.	✘	Mari studass	Teaching assistant
<input type="checkbox"/>	studass5	March 2, 2020, 2:04 p.m.	✘	Martin studass	Teaching assistant
<input type="checkbox"/>	studass2	March 2, 2020, 2:02 p.m.	✘	Pål studass	Teaching assistant

FILTER

By is admin

All
Yes
No

By is active

All
Yes
No

By role

All
Student
Teaching assistant
Instructor

Figure 40: Admin panel, user overview

In figure 40, we can see all the users in the database, and do certain actions.

The screenshot shows the Django administration interface for a user. The page title is "Django administration" and the user is logged in as "ADMIN ADMINSEN". The breadcrumb trail is "Home > User > Custom users > Alf ElevG Christensen". The main heading is "Change custom user" with a "HISTORY" button. The form contains the following fields:

- Username:** alf_elevg
- User_id:** 18a1ec21-1aa4-45b5-a168-61eec4293e30
- Password:** algorithm: pbkdf2_sha256 iterations: 180000 salt: cQhw0K***** hash: v6aQaO*****
- Name:** Alf ElevG Christensen
- Role:** Student
- Selected subject id:** 1
- Email:** alf_elevg@feide.no

Below the form, there are two sections:

- Permissions:** A checkbox for "Is admin" is currently unchecked.
- SCORES:** A table with one row for "Score: Alf ElevG Christensen" with a "Delete" button. The score is 2, the team is "Øvingsgruppe 1", and the date registered is "2020-02-27".

Figure 41: Admin panel, view and change one user

Figure 41 shows an example of clicking into one user in the database. Here it is possible to change the user and save it, including deleting it.

7.3 Qualitative user testing

7.3.1 Test form

In order to try to verify and find out if the concept of TeamAccelerator is of value to students and course staff, some tests were conducted on various people. Due to the situation with the COVID - 19 virus, the tests needed to be conducted over the internet, either in the form of video or just written. Ideally, the tests should have been conducted with the test - subject in-person, but this was not possible. The user-testing was conducted as follows:

- The test subject was firstly presented with the basic concept of the application, and the main problem it is trying to solve.
- After this, the test - subject was presented with several tasks in chronological order, to complete one after another.
- These tasks are presented below with the application introduction

7.3.2 Introduction for student test

Below is the introduction the test-students was presented with:

"This application is about teamwork at NTNU in project based courses. The goal of the application is to make it easier to prevent bad or incomplete group work. The application is relevant for you as a student that at this point in time, or in previous semesters has worked with project based courses in teams. The app consist of a simple platform where you have the ability to give feedback to your project team every week, based on how happy you are with your group. This feedback is a number from 1 to 5, where 1 is very bad and 5 is very good. In other terms, 1 means that you are very unsatisfied with your team, and 5 means the opposite.

The other students on your team will not be able to see what you have answered, but all students can see the average score of their team. Professors, teaching assistants and other course staff will get another side of the application where they have overview over all the groups, the average-rating of every group, and all individual scores of every student. The goal is to find out if students like you feel that this can be a valuable tool that can help improving group work, and make it easier to solve problems related to it.

Your honest feedback will be very appreciated. Don't be afraid to point out errors, drawbacks, weaknesses or other things you think should be improved or added. The more feedback the better! Think "loudly" and note everything that is of value."

7.3.3 Student test tasks

The test subjects went through the following tasks:

- 1. You are registered in the subject TDT4140 - Software Engineering at NTNU. This is a project based course where the students work in teams on a project that lasts the entire semester. The project counts 100% of the grade, so it is important that you are satisfied with your team. Teamwork can be challenging from time to time, therefore, the professor of TDT4140 has recommended a new application where you as a part/member of your team, can give weekly feedback in the form of a number from 1-5, where 1 means you are very unsatisfied and 5 means you are very satisfied with your team. You now wish to log into the application through your phone.*
- 2. Lately, you have felt that your team is doing bad. You are not satisfied with either the progression or the other team members. You want to give your team a bad rating this week.*
- 3. You don't remember which student assistant that is responsible for your team, and you wish to find this out now*
- 4. During the project period, you have given different scores to your team the different weeks. One of the weeks, you gave a very bad rating, while the other weeks, you rated your team pretty good. You don't remember what weeks you gave the bad rating, and you wish to find this out now.*
- 5. Since you are not happy with your team this week, you wish to contact your teaching assistant.*
- 6. You want to view your own profile in the application, and wonder if there is any more information about the app.*

7. You are done for this week, and now wish to log out of the app.

7.3.4 Introduction for staff test

Below is the introduction the staff that was tested was presented with:

This application is about team work at NTNU. It is trying to make it easier to prevent bad and incomplete team work. Bad teamwork in this case is teams that have one or several members that are unsatisfied with their team, and feel like the team does not achieve any or enough results, which can lead to a bad grade or in the worst case failing the course. This application is relevant for you as a professor or teaching assistant, that is currently leading and assisting a project based course on NTNU, or have done so in the past.

The app consists of a simple platform where the students give feedback to their teams based on how satisfied they are with the teams. This feedback is for each student a number between 1 to 5, where 1 means that the student is very unsatisfied and 5 is very satisfied. You as a professor or TA, will get another side of the application where you have overview of all the teams and the individual team members ratings and average ratings. The goal is to find out if this is a tool that can enhance and improve the team work and make it easier to solve problems that occur during the project period. This is especially relevant for teams that have a poor performance.

Your honest feedback will be very appreciated. Don't be afraid to point out errors, drawbacks, weaknesses or other things you think should be improved or added. The more feedback the better! Think "loudly" and note everything that is of value.

7.3.5 Staff test tasks

1. Log into the application with FEIDE.
2. The app points out teams that has an average rating below 2.5. You now want to find out which teams are below this point.
3. One of the teams with rating below 2.5, you wish to follow a bit closer. Therefore, you want to "pin" team 16, so that this group will appear at the top of the list over all teams later.
4. You are responsible for 2 teams, Team 23 and Øvingsgruppe 1. You now want to view more information about Øvingsgruppe 1
5. An average score of 2.9 you think is pretty bad. Du wish to see if most people on the team have given more or less the same rating, or if there are a large gap between the members.
6. You now wish to view more detailed information about Øvingsgruppe 1.
7. One of the weeks, Øvingsgruppe 1 had the lowest average rating. You now want to find out which week this is, and view the different individual member ratings of this week as well.
8. You feel like you have gotten a good overview, and wish to continue over to the list of all teams in TDT4140
9. You no longer want to have Team 16 at the top of your list. Do something about it.
10. To gain a better overview, you now wish to sort the list based on rating, in an ascending order.

11. *At first glance, you cannot see Team 14 in the list. Therefore, you wish to search for team 14 and view some more info about it.*
12. *You want to view your own profile in the application, and wonder if there is any more information about the app.*
13. *You are done for this week, and now wish to log out of the app.*

7.4 Qualitative user testing results

The results and feedback from the various people tested on was valuable. Some test users had a lot to say, others less. In general, most of the test subjects thought the concept of this application was interesting and could have potential. The test subjects are mainly students that are both studying and working part time as teaching assistants, or have done so in the past. All of them have experience with teamwork. The age of the people tested ranges from 22 to 26. Displayed below are the feedback and results of testing the application on these people.

7.4.1 Student tests

First impression	<ul style="list-style-type: none"> ● <i>"First impression of the application was great. The application was in general clear and easy to understand".</i>
Difficult to understand	<ul style="list-style-type: none"> ● Task 4: When finding the previous scores, it took a couple of seconds before he understood that this was on the same page as the rating. ● The first reaction was to click into the profile page instead ● After a couple of seconds, he understood where he could find the previous scores
Things that can be added	<ul style="list-style-type: none"> ● On the profile page, the "More info" button can be expanded by default
Errors	None
General conclusion	In general, he thinks the concept of this application can be very useful. Himself personally have experienced project based courses where they have done similar ratings, just in the form of physical papers. He thought it would be great to have this digital instead.

Figure 42: Result student 1

First impression	<ul style="list-style-type: none"> • <i>“Minimal effort and potentially a great help for students”</i>
Difficult to understand	<ul style="list-style-type: none"> • Task 5: If this hadn't been a task, he would have assumed that the “contact” button was to contact the system responsible for bugs and errors, not contacting the responsible teaching assistant.
Things that can be added/improved	-
Errors	None
General conclusion	<ul style="list-style-type: none"> • In general, all the tasks were easy to complete, and it was easy to find what he was looking for.

Figure 43: Result student 2

First impression	<ul style="list-style-type: none"> • <i>“Cool application with an interface easy to understand”.</i>
Difficult to understand	<ul style="list-style-type: none"> • Task 6: It was strange that the extra information about the application, the “more info” button, was in the profile page of the app.
Things that can be added/improved	<ul style="list-style-type: none"> • Place the “more info” section on the profile page on a separate tab or somewhere else.
Errors	None
General conclusion	<ul style="list-style-type: none"> • Same as first impression

Figure 44: Result student 3

First impression	<ul style="list-style-type: none"> • <i>"Cool app, but I tested it on pc instead of phone, and it is not working that well on pc. To much space, since it is mainly made for phone"</i>
Difficult to understand	None
Things that can be added/improved	<ul style="list-style-type: none"> • Task 2: It was difficult to slide the slider between 1 to 5. It should be very easy to slide with the finger. • It should not be necessary to log into the application every time visiting the URL. It should be auto-login if the user has already logged in.
Errors	None
General conclusion	<ul style="list-style-type: none"> • Same as first impression

Figure 45: Result student 4

7.4.2 Staff tests

First impression	<ul style="list-style-type: none"> • <i>"The application has potential. If the students start to use this at the very start of the semester, I think certain problems with teams that arise can be handled."</i>
Difficult to understand	<ul style="list-style-type: none"> • Task 7: After clicking to view the scores of one week, the date text is a bit confusing. • On the same task, it is confusing when you click on one week by the rows of another week appearing under.
Things that can be added	<ul style="list-style-type: none"> • On the modal that shows the information about one group, the email of the responsible TA should also appear. • Task 7: All other weeks should be hidden when you have clicked on one week, to avoid confusion. • Status in the list header should be renamed to rating
Errors	None
General conclusion	Same as the first impression.

Figure 46: Result staff 1

First impression	<ul style="list-style-type: none"> • <i>"From my experience with teamwork, there are always teams with one or several people that only free-rides and don't contribute. This application may have potential in solving some of the problems related to this".</i>
Difficult to understand	<ul style="list-style-type: none"> • Task 10: It is not very intuitive to know that it is possible to sort the list. If this was not a task, it would be more difficult to know where to click to activate sorting. It would have helped if the list was pre-sorted, such that that the arrow indicating that sorting is on, also is indicating that it is possible to click. • Not 100% intuitive that pinned elements appear at the top of the list
Things that can be added/improved	<ul style="list-style-type: none"> • Task 7: It was easy to find the specific week and the individual scores, but he thought it would be possible to click on the graphs, which dimmed when he hovered over them.
Errors	None
General conclusion	<ul style="list-style-type: none"> • This could absolutely be a useful tool to deal with team work problems. But, what motivates the students to go in and rate weekly?

Figure 47: Result staff 2

First impression	<ul style="list-style-type: none"> • <i>"I like the concept. It would be interesting to see if it worked in a real situation"</i>
Difficult to understand	<ul style="list-style-type: none"> • Task 2: Team below 2.5 in rating should have an arrow pointing down when it is not expanded, and the opposite when it is. • Task 10: There should be some kind of indication or marking of the headers, that indicates that it is possible to sort the list.
Things that can be added/improved	None
Errors	None
General conclusion	<ul style="list-style-type: none"> • Same as first impression

Figure 48: Result staff 3

First impression	<ul style="list-style-type: none"> • <i>"This is quite easy to understand"</i>
Difficult to understand	None
Things that can be added/improved	<ul style="list-style-type: none"> • Task 3: Pinning the group does not make the group appear at the top of the small list of teams below 2.5. • No purpose with arrows under the expanded categories. • The arrow in the right corner of the categories should be switched up when opening the category • Task 10: It should be possible to click on the arrows in the headers, not only the text. It also felt unnatural that the pinned elements were sorted by themselves. At first glance it looked like an error in the sorting. Sorting of pinned elements and other elements should be separated, or all should be sorted together.
Errors	None
General conclusion	<ul style="list-style-type: none"> • Easy to understand and use, with some exceptions.

Figure 49: Result staff 4

7.4.3 Elements changed based on the user testing

Elements in the application that was improved and changed based in the user testing:

- **Result staff 1:** When viewing the history ratings of one team, the date text of one week was made a lot clearer. The week number was also added.
- **Result staff 1:** When clicking on one history row to view individual scores of a team, all the other weeks should hide to avoid confusion of what data belongs to that week. This is implemented.
- **Result staff 1:** When clicking on one team to view information, the email of the responsible teaching assistant should be displayed.
- **Result staff 1:** In the list of all teams, the header of the list named "Status" should be renamed to rating, which is now the case in the app.
- **Result staff 4:** Removing the unnecessary arrows below the categories when they are expanded in the overview section.
- **Result staff 4:** Both the pinned and unpinned items are now sorted together, instead of sorting them separately in the same list.

7.4.4 Elements that can or should be changed based on the user testing

Elements in the application that can be improved based on the user testing:

- **Result student 2:** Rename or move the tab for contacting the responsible teaching assistant, to avoid confusing this with contact of system responsible for bugs and errors.
- **Result student 3:** In the information section of the application, remove the "More info" button and place it on a different tab or another place.
- Improving the PC version of the application. The app is tailor made for smartphones, but not desktop. Re-Arranging the whole PC app to make it look more like a desktop application.
- **Result student 4:** Improving the slider in the student rating page. Make it easier to slide.
- **Result student 4:** Implement auto login when visiting the base URL. Auto login on all the other URLs is already implemented.
- **Result staff 2 & 3:** Make some kind of indication that makes it easier to understand that it is possible to sort the list over all teams. A possible solution is to pre-sort the list so that the indicating arrow appears by default.
- **Result staff 2:** Make it easier to click on the graphs. This was possible the whole time, but it may not work as well in some browsers, which may be the reason it did not work in this test case.
- **Result staff 2 & 3:** The arrow in the right corner of the categories should be switched up when expanding the category.

7.5 Quantitative user testing

To try to test the application in a real environment, it was released to a real course at NTNU, TDT4180 - Human Computer Interaction. As described earlier, due to large delays with deployment, and the COVID - 19 situation, the data gathered from this testing is rather incomplete, but some results were achieved.

- In total, there were 61 users registered in the application from TDT4180.
- Out of these users, 45 of them rated their team.
- Nearly all the users only rated their team once.
- In nearly all teams in which someone has rated, there is only one person at the team that rated.
- There were only 5 teams that had several team members rating the team.
- Only 2 teams had more than 2 people rating the team.
- Only one team where several people rated, had one person rating 2 different weeks.
- Out of 153 teams in total, only 34 teams had someone that rated it.
- Out of these 34 teams, the total average rating was 3.5 out of 5.

7.6 Conclusion & future work

There are certain things that can be improved and further extended in the application. The following list provides further extensions and improvement of the application:

- Fetch the information about what course people are enrolled in from blackboard API, instead of having to upload this manually in the database.
- To further emphasize the gap in rating between the students, make it more visible with graphical representation.
- Based on the user testing, the elements pointed out that was not implemented or improved, should be improved or implemented.
- Implement the rest of the functionalities in the design sketches. These can be found with images and descriptions under section 3, and in the appendix.







Another important future work is to publish the application in a real course at the very start of a semester, under normal circumstances without COVID - 19 and the restrictions that follows from it. In this master thesis and testing, the application was not mandatory to use, and students tend to not do the effort of doing extra work if they don't get anything back from it. The same applies to teaching assistants. If this application was counting 1-5% of the grade, there is a probability that a substantial larger amount of students would use this application. But, that is up for the future testing to prove.

Based on the work of this thesis, there are several indications that this application and way to improve team work, could be useful in many situations. Most of the students that tested the

application, thought the concept was interesting, and could be used to potentially prevent bad team work. But, the approach on how to get students to actually use it, need to be further developed. Students usually do not do academic related work voluntary, if the work does not have an impact on any grade. The approach described above could be a potential solution to that. Overall, this application could be used today in real academic situations, but ideally, the application needs to be tested and potentially improved more, to gain a greater verification that it actually prevents bad team work or at least improves some aspects of it.

8 Resources used

The various images and icons used in the design of this application are gathered from <https://www.flaticon.com>. This includes the following icons:

- Emojii icons used in the rating slider - made by "Smashicons". 
- By Smashicons: pie-chart, add and pin 
- By Freepik: status-bar, info button, group, mortarboard, professor 
- By Itim2101: Book 
- By Gregor Cresnar: Avatar, teamwork 
- By Becris: Meeting 

All the other icons used in both the design and the application itself are self-designed and made. This also includes the application logo.

References

- [1] N. Bendaly. *6 practices to help build a healthy team climate*. 2018. <https://www.theladders.com/career-advice/6-practices-to-help-build-a-healthy-team-climate>.
- [2] A. Burke. *How to use groups effectively*. 2011. https://uncw.edu/jet/articles/vol11_2/burke.pdf.
- [3] P. Chang, Y. Brickman. *When group work doesn't work: Insights from students*. *CBE Life sci. Educ.* 17. 2018. <https://doi.org/10.1187/cbe.17-09-0199>.
- [4] P.T Coleman. *The Science of Teamwork*. 2018. <https://www.psychologytoday.com/blog/the-five-percent/201806/the-science-teamwork>.
- [5] T.E. Dybå T. Haugset B. Lindsjörn Y. Dingsøy, T. Fægri. *Team Performance in Software Development*. 2016.
- [6] Charles. Duhigg. *Project Aristotle*. 2016. <https://rework.withgoogle.com/print/guides/5721312655835136>.

- [7] L.S Huang. *Group work strategies to ensure students pull their weight.* 2018. <https://www.facultyfocus.com/articles/effective-teaching-strategies/students-riding-coattails-group-work-five-simple-ideas-try>.
- [8] C. Lino. *The Psychology of Teamwork: The 7 habits of highly effective teams.* 2016. <https://positivepsychology.com/psychology-teamwork>.
- [9] University of Queensland. *Problems associated with group work.* 2014. <https://www.uq.edu.au/student-services/learning/problems-associated-group-work>.
- [10] P. Brame C.J. Wilson, K.J. Brickman. *Group work.* 2018. <https://doi.org/10.1187/cbe.17-12-0258>.

Appendices

A The old login process

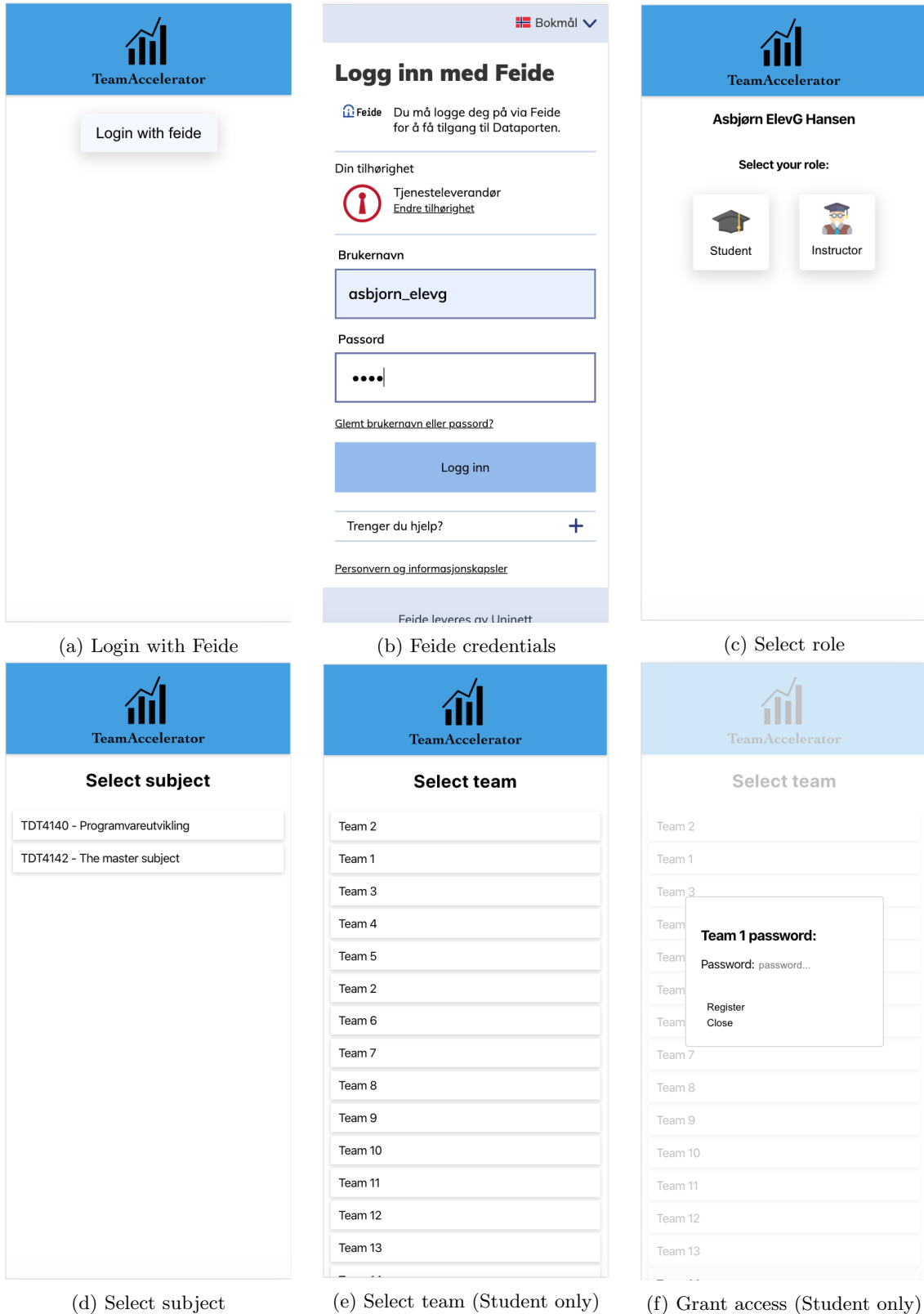
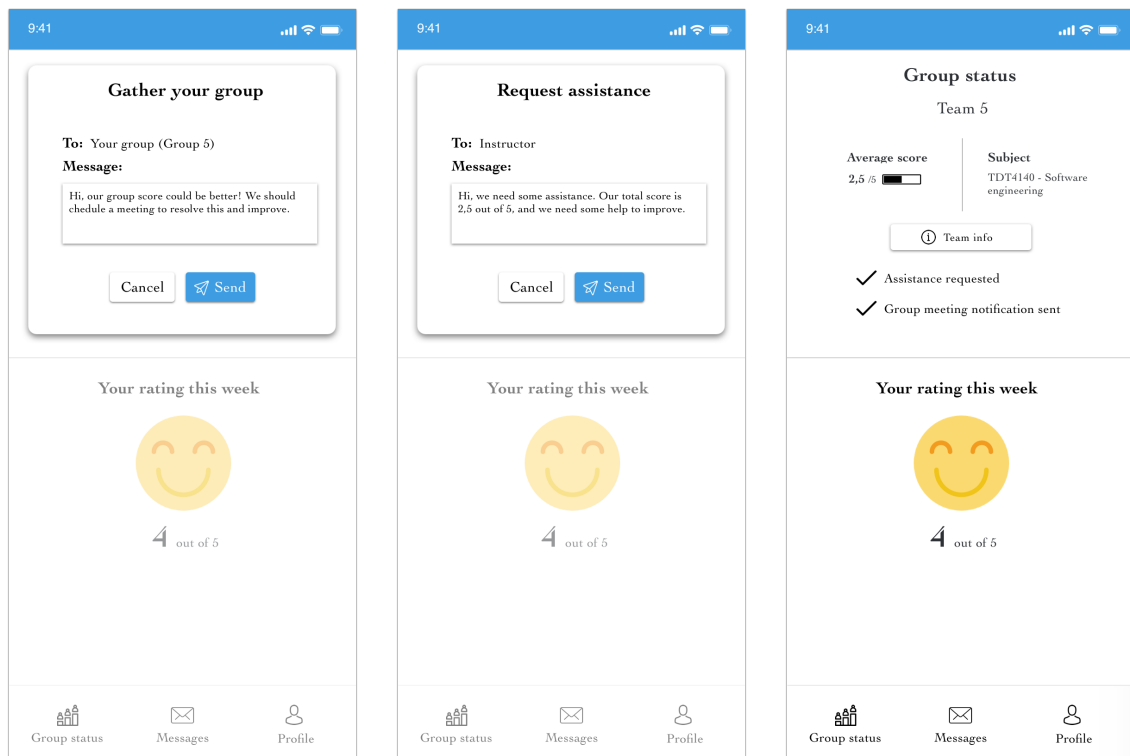


Figure 50: Appendix: Login process

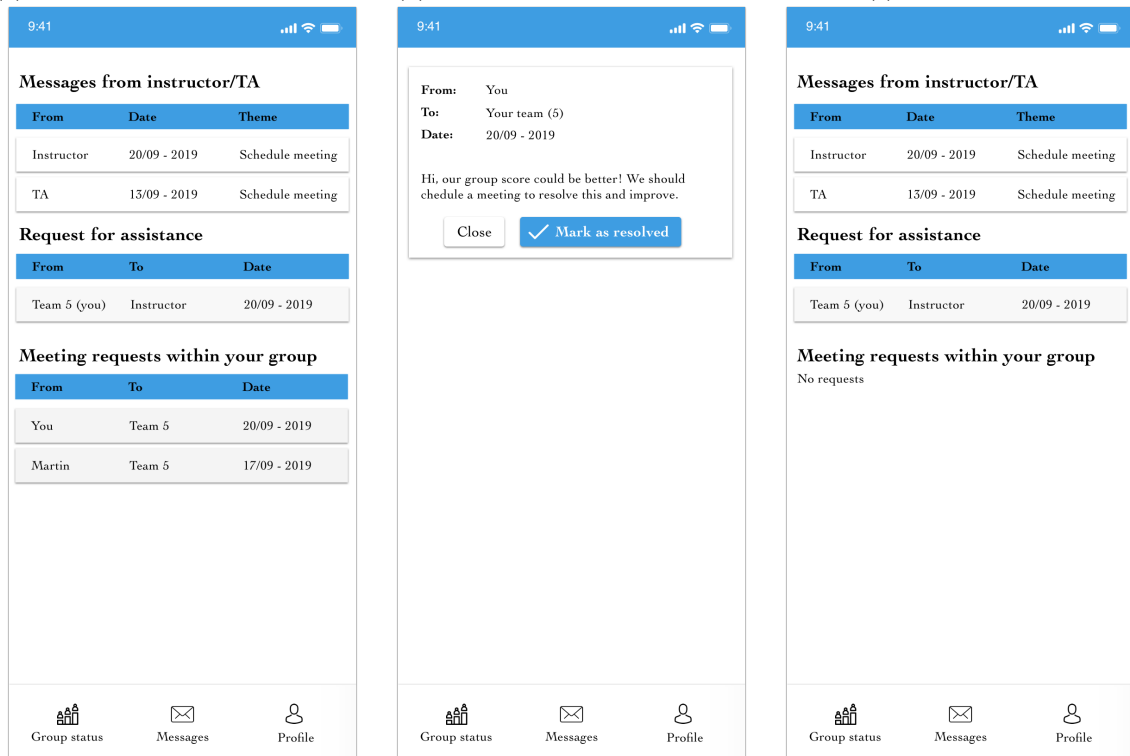
B Additional and alternative design



(a) Gather the group for a meeting

(b) Request assistance from staff

(c) After requests

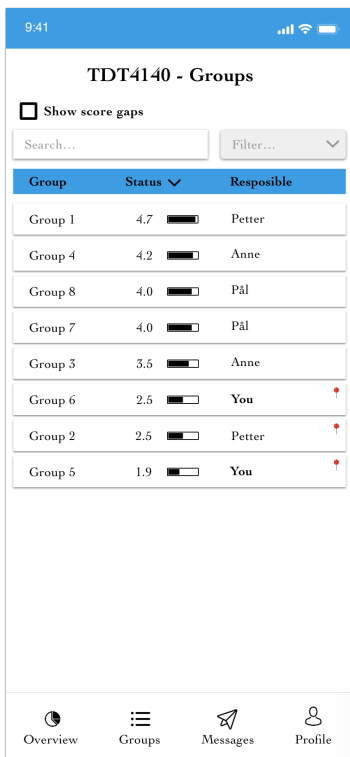


(d) Messages

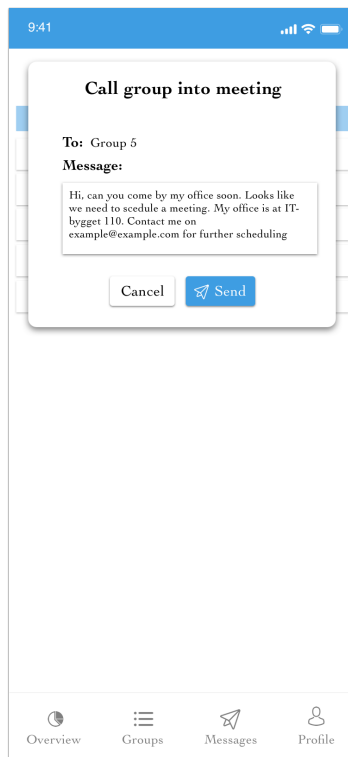
(e) One request

(f) After request resolved

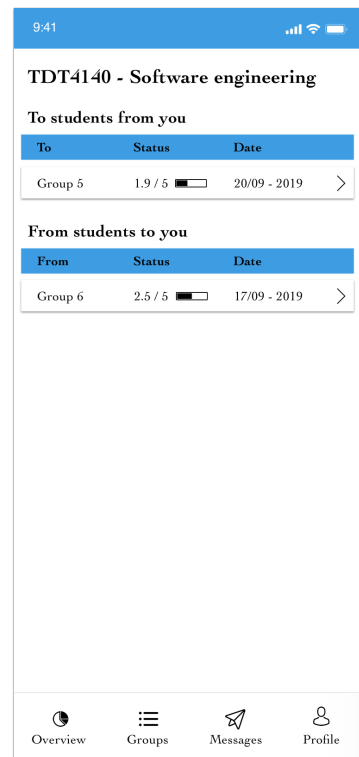
Figure 51: Appendix: Additional design of student side



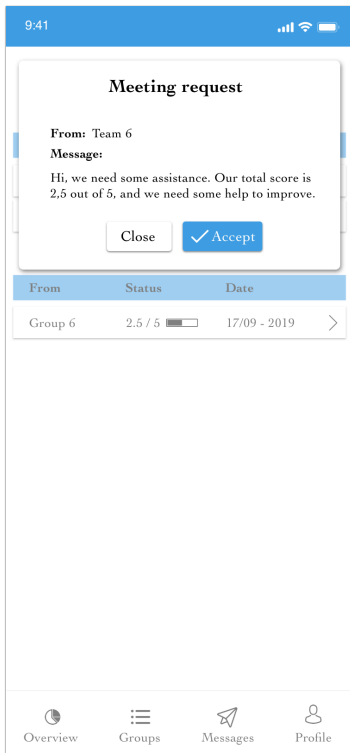
(a) Having additional filters



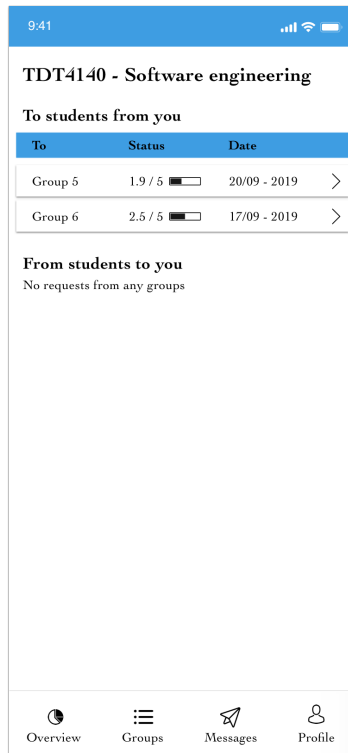
(b) Call group into meeting



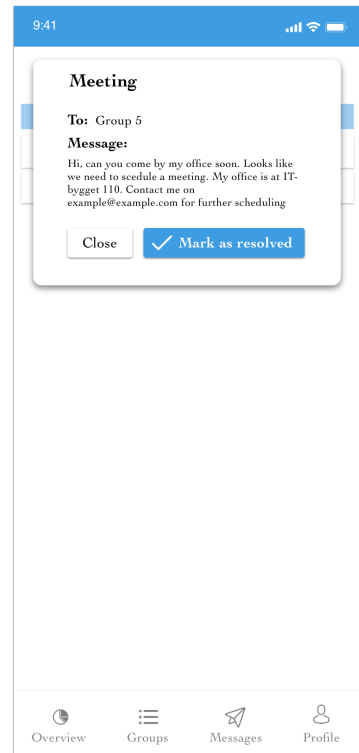
(c) Messages



(d) View a request from a team



(e) After request accepted



(f) View a message from a team

Figure 52: Appendix: Additional design of staff side

C Personas

View the personas in the attachments, or simply click the links below.

- Persona of a typical professor user: **[Click here to view.](#)**
- Persona of a typical teaching assistant user: **[Click here to view.](#)**
- Persona of a typical student user: **[Click here to view.](#)**

