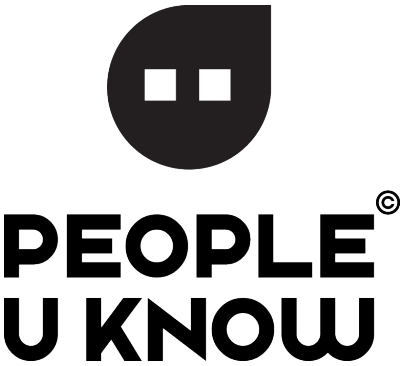
NTNU – Trondheim
Norwegian University of
Science and Technology

TDT4290 Customer Driven Project

PeopleUknow report



Group 10:

Kristoffer Finckenhagen
Ivar Helland Hessevik
Christian Barth Roligheten
Vigleik Lund
Matias Halsteinli Unsvåg
June Kieu-van Thi Bui

November 11, 2016

Abstract

Some students struggle with the social aspects of school. Lack of motivation, loneliness and bullying creates undesirable environments in a classroom. PeopleUknow is developing a digital toolkit used to improve social skills. The project aims to create well-being, introspection and motivation in a classroom through social initiatives.

This report was written during TDT4290 Customer Driven Project, fall 2016. The purpose of this report is to document the working processes, as our group worked along with PeopleUknow to create the digital toolkit. The project was a software development project for the company PeopleUknow. Since they are a start-up company our group was involved in several different activities, including idea-creation-workshops, graphical design, user testing, and software implementation.

The product was a mobile application which was intended to be used by kids and teachers in school. Since the most popular mobile operating systems nowadays are iOS and Android, some of the developers hired by PeopleUknow had decided that the best way to create this app was by creating an iOS/Android hybrid. This allowed us to create a cross-platform application using mostly JavaScript and CSS.

Even though our project included a variety of tasks our main task was to create and implement a prototype of an MVP (minimum viable product) that PeopleUknow could show to potential customers.

1. Preface

We would like to thank our advisor Soudabeh Khodambashi for the continuous feedback she has given us through the project.

We would also like to thank the Marianne Johnsen, our contact at PeopleUknow, and the rest of the team for their input during the development-process.

Contents

List of Figures

List of Tables

2. Introduction

In this chapter we will describe the overall context of the project. This includes the description of the project, the people who are involved in this project, and the background and goals of the project.

2.1. Project Description

Our project is called "A digital tool to improve social skills & learning in the classroom". The scope of this project is to create a prototype for a digital toolkit that teaches social-competence in classrooms. The prototype will take form of a mobile app for Android and iOS. The core of the project is to create a platform which facilitates social skills. The project will focus on implementing features that can later be developed into an MVP of the toolkit, and be used for demonstration to potential customers and investors.

2.2. Involved Parties

In this section, we will describe the parties that have been involved in our project. Our main stakeholders in the project is the development team, PeopleUknow and the end-user. At the end of this section, Table 2.2 lists some of the stakeholders and their most important criteria of concerns for the project.

2.2.1. The Development Team

Our team consists of six 4th year students currently taking a master's degree in computer science at NTNU.

All members of the team have a broad range of work preferences and experience; with some preferring coding and others more comfortable with design and testing. Some also have experience from internships and part-time jobs within the IT industry, while others have worked as student assistants at NTNU. Team roles were assigned according to preference and past experience.

The Table 2.1 describes competences and experience of each member. Most of us have experience with Scrum and different programming languages.

Table 2.1.: Group Competences

Name	Relevant Competences and Experiences
Kristoffer	Scrum, Kanban, Android Development, Java, Python, MySQL, MatLab
Matias	Scrum, Android Development, User testing, Java, Python, MySQL
Ivar	Scrum, Android Development, Javascript, HTML, git, Java, C, Python, MySQL, TDD
Kieu-van	Scrum, Java, Matlab, Design Thinking, User testing, Project management, LaTeX
Vigleik	Scrum, Java, Matlab, Python, .NET, SQL
Christian	Kanban, Project Managment, Javascript, Java, C, C++, Python, HTML, CSS, Git, AngularJS

2.2.2. Customer

PeopleUknow [34] is a start-up company whose vision is to teach children to find a balance between asserting themselves and caring for others. The company currently consists of a small group of people sharing that same vision, and they are constantly adding new members to their team. Their current team consists of designers and teachers. The company is currently looking for funding and support for developing their ideas further.

Marianne Johnsen is our contact at PeopleUknow. She was the one that gave us feedback and ideas on how to develop the prototype. Marianne is also the product owner, i.e. the CEO. Her responsibilities as product owner in this software development project includes making sure that tasks in the backlog have a user centered focus, and are not too technical, as well as making sure all requirements from the customer are represented [20].

Students

PeopleUknow hired two students during the summer for developing the application. They have knowledge about PeopleUknow's codebase and were giving us useful support during the early stages of development.

2.2.3. Webstep

Webstep is a software consulting firm based in Trondheim. They were giving technical guidance to PeopleUknow. They have knowledge about similar projects and they were giving us support regarding technical issues. For this project, we had some meetings with them and they offered us help by giving us a technical person we could ask if there was a problem.

2.2.4. End-user

By end-user, we mean students and teachers. They are the ones who will eventually use the product. It is therefore very important to understand them and get feedback from them.

2.2.5. The supervisor

We also had a supervisor, which was responsible for giving feedback on the project and answering questions if the group had some questions. Soudabeh Khodambashi was our supervisor. She guided us through the weekly meetings and helped us to get on the right track.

Stakeholders	Criteria of Concerns
Development team	**Readability**: The code should be commented and structured well to make it easy for all group members to understand each others code. **Modifiability**: The architecture that is used should be appropriate for the app, to make it easier to change or add new features. **Version control**: The developing team should use a framework for version control, e.g. Git, to make sure work is not lost in case of accidents. By using version control it is also possible to go back to previous stages of the system if we create functionality that we realize is undesired or different from the preferred result
End user	**Usability**: The app must be interesting and easy to use. If teachers find the app cumbersome and struggle to use it, they will most likely not introduce it to their class. **Entertainment**: The app should be fun to use because the more the students use the app, the bigger the improvement in social skills and well being will be.
Other developers at PeopleUknow	**Readability**: The code for the app should follow standards for commenting and documenting, to make it easier for different developers to complete or extend the code. It will also make it easier for different developers to work in parallel. **Work with React**: The app should be able to test for either iOS or Android using React Native, to make it easier for the other developers to evaluate our work, and continue their work accordingly
PeopleUknow	**Functionality**: The app should contain the functionality they would like the app to have, within reasonable limits. **Security**: The app should protect the information of users. **Modifiability**: As this is an MVP and not a final product the app should be easy to modify, and/or add new functionality later.
Webstep	Overall success of the product.

Table 2.2.: Stakeholders and Criteria of Concerns

2.3. Project Background

According to our customer [34], the background for this project is that many students are lonely, unmotivated or bullied. This creates much stress and it worries them.

In today's school system with a lot of curriculum to rush through, the school focus on getting through the curriculum. Other important things such as introspection and social competency is not prioritized as much. Many students don't even know all the people in their class, which can make them feel unsafe in their own class environment.

2.4. Project Goal

The goal of this project with PeopleUknow is to help students to get to know their classmates. By developing an application which provides the students with digital tools, we are hoping that the students will be using it in class to learn more about themselves and their classmates.

For this project, our specific goal was to develop an application which comes with five main features: Home, Login, GetToKnow, Exercise, and Scrapbook. GetToKnow feature will make it possible for each student to get new group partners, and get to know them. With the group each student got the Exercise feature contains a set of exercise that the group is supposed to do. When a group has completed an exercise, they have the option to share their experiences with their class.

Another main goal of this project is to get a real life experience on how software development works, such as working in teams, how to manage the information, what technologies to choose, and working with a real customer.

2.5. General Terms

The application is intended to be used by students and teachers at lower secondary school (ages from 13-16) and upper secondary school (ages from 16-19).

2.6. Scope

The project started 23th August and it will end on 17th November with a presentation of the result of the project. The report of the project will be delivered 11th November. Therefore, the estimated amount of work is calculated for the period between 25th August to 17th November.

2.7. Chapter Outlines

This section describes how the rest of the report is organized.

Chapter 3 - Pre-study

Chapter 3 is about the information that has been gathered in order to understand the total problem we are solving.

Chapter 4 - Planning

Planning chapter describes how we organized our project, documentation and management tools we used, and describes the choice of lifecycle-model. The last section of this chapter describes the risks for this project.

Chapter 5 - Requirement Analysis

Requirement Analysis describes how we identified the requirements; both functional and non-functional requirements.

Chapter 6 - Quality Assurance

QA chapter describes the programming environment, coding standard and code review procedure.

Chapter 7 - Technologies

Chapter 7 describes the different technologies and tools that we used for the development of the project.

Chapter 8 - System Architecture

Chapter 8 presents the design of the system, and describes the architecture we implemented.

Chapter 9 - Software Security

Software security chapter describes the security aspects of our project.

Chapter 10 - Testing

The testing chapter describes the various testing methodologies used and considered in the development of the application.

Chapter 11-14 - Scrum-sprints

Chapter 11-14 describes each of the four sprints in detail.

Chapter 15 - The Final Product

The final product is presented in chapter 15.

Chapter 16 - Evaluation

The final chapter presents the reflection on the project work and discusses what went well, what could be improved and further work.

3. Pre-study

This chapter describes the information that has been found in order to understand the total problem we are solving.

3.1. Problem Description

22% of the students in 2013 say they have been bullied during the school year [24]. Bullying can lead to loneliness, lack of motivation and/or suicide [5], which are really serious issues. One reason for bullying is that students are lacking of understanding each other [30], which can lead to judgement among the students . A study performed by Yale University, discovered that bullied children were two to nine times more likely to develop suicidal thoughts [33].

In order to reduce this problem, PeopleUknow wants to create an application to reduce the lack of motivation, loneliness, bullying and expectation press, make students care about each other, and create a safe and positive class environment. The application will teach them more about self development and learn about situations related to social competence.

We wanted to help PeopleUknow to develop an application to provide students with digital tools, which can help students to get to know each other in their class, learn more about themselves, and learn and practice in social situation.

3.2. Current System

PeopleUknow has focused on design of the app and research. They have made sketches of key aspects of solution and are doing research on how to best motivate children. During the summer they hired two students to begin work on the coding of the app. They mainly focused on design sketches, but did some initial work on the implementation. The students chose React Native [10] as coding framework and Firebase [17] as backend-database. They have created a Git repository and have started experimenting with the framework. The code in the repository seems of little value to us and is not likely to be used further in development.

The provided design sketches described ideas of the layout of each screen and how different functions could be implemented. The sketch for the main components can be seen in Figure 3.1.

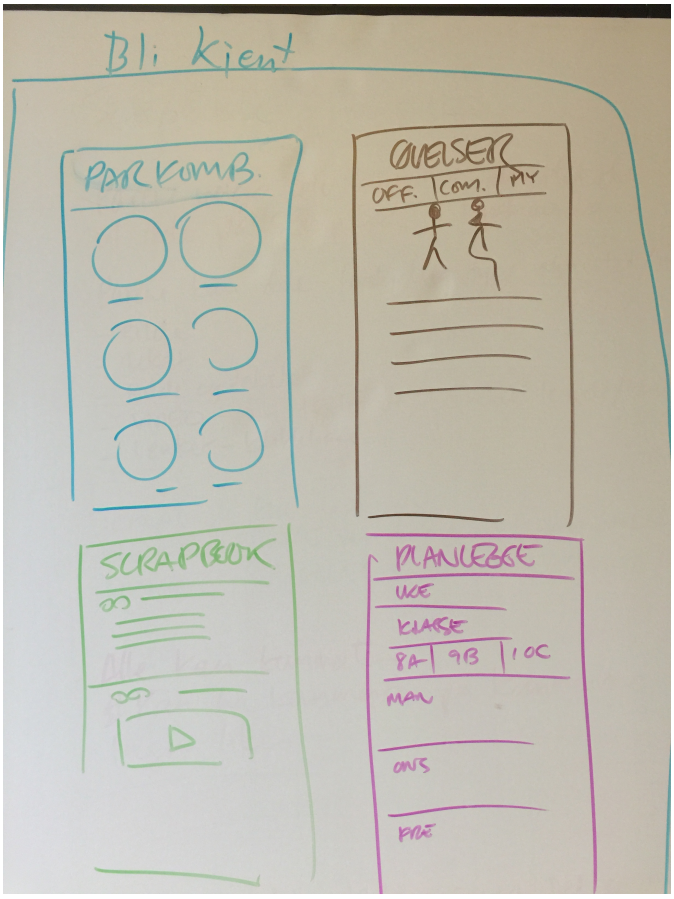


Figure 3.1.: Sketches of the main functions.

3.3. Project Solution

This section looks at the expected solution and the possible solutions. In 3.3.3 we tried to understand our users through creating personas.

3.3.1. Expected Solution

The desired outcome of the project, as described by PeopleUknow, consists of five main parts:

- Scrapbook: A feed that displays activity within a class. It is intended that classmates can share pictures, text etc. with each other.
- GetToKnow: An algorithm to divide a class in groups of two or three students each week. The algorithm should avoid grouping students that has already been in groups. The teacher should be able to rearrange group members.

- Exercises: Each group can be given a social activity which they must do together. A video explaining the task may be given. The teacher can choose which activity to do and plan when it should be done.
- Home: A frontpage where students can navigate.
- Login: A way to login the students or teachers. To easily interact with schools, PeopleUknow were considering integrating the app with Feide [13].

PeopleUknow has informed us that their ideas are not rigid and that they might change their mind.

3.3.2. Possible Solution

While there has been done some initial work on the project, these are mostly sketches. Therefore, it could be possible to start from scratch and use a different framework or programming environment. As mention previously, React Native was used by the summer students. They suggested that we used the React Native framework, in combination with Firebase on our project. This suggestion was later endorsed by Webstep, which led us to our decision to continue with those frameworks.

It would be possible to create the app using a different framework than React Native. Many frameworks provide similar functionality to React Native, including camera access and cross-platform support. Xamarin[49] and Cordova[2] are similar to React Native. However, we do not have any experience with these framework. By using the existing solution, we can ask for help when we encounter a problem.

An alternative could be to create the app as a web page instead of an app. This could make it easier to distribute and future implementation with desktop/laptop interface. However, camera and other native components on phones are not easily accessed with a web page.

Implementation of the solution is stated in the previous section. The main possibilities are connected to layout and look of the app and will likely be incrementally improved through development and feedback from the customer. However, login with Feide is something that would be nice to have, but is not essential for a working prototype. Such integration would take a lot of time and the effort could be spent better elsewhere. We could design the login system such that integration with Feide is possible in the future.

3.3.3. Personas

In order to understand our users we have created some personas. The people we have describes in this section are fictitious. Figure 3.2 and Figure 3.3 describe two types of characters that will use our application: student and teacher. The template we used was from Xtensio.com[19].

"I wish I knew someone I could eat my lunch with."

Age: 16
Work: Student
Family: Separated parents
Location: Trondheim, Norway

Goals

- Want to pass upper secondary high School
- A task that needs to be completed.
- Get a good job.
- Get some new friends.

Bio

Emma is a student in upper high school, she takes classes in scientific major and in her spare time she likes to play online games. Last year her mother and her father got divorced. Emma and her mother moved therefor from Melhus to Trondheim. In Trondheim, Emma knows only her grandparents and her neighbors.

Since Emma don't know anyone at her age in Trondheim, she prefers to stay at home.

Frustrations

- Difficult to talk to people.
- Don't know her classmate.
- Don't have anyone to eat lunch with at school

Social Media channels

Facebook

Youtube

Twitter

Instagram

Personality

Introvert — Extrovert

Analytical — Creative

Conservative — Liberal

Passive — Active

Figure 3.2.: Personas: Emma Nord

"I love seeing the smile on each students when they understand."

Age: 46
Work: Teacher
Family: Married, 2 kids
Location: Trondheim, Norway

Goals

- Be the best teacher in history.
- Teach his major in a way it is exciting for the students.
- Wants every student to understands math.

Bio

Anton loves his job as a teacher. He likes to teach the student about history and math. Anton believes in active learning so he likes to group students into groups so they can solve and discuss problems together.

When Anton is not working, Anton is at home helping his kids with math. He believes also that it's important that his children get to know and play with other children.

Frustrations

- Some students are unmotivated.
- Dividing the class into pairs.
- Want everybody to be nice to each other.

Social Media Channels

Facebook

Youtube

Personality

Introvert — Extrovert

Analytical — Creative

Conserva — Liberal

Passive — Active

Figure 3.3.: Personas: Anton Hagen

3.4. Evaluation Criteria

For this project the customer wanted us to focus on the design and the functionalities A.1. In today's society there are a lot of applications out there, some of them are good and some are bad. An application with a good concept can be bad for instance if the design is crappy. If the application does not give the user a good user experience, then the potential users will not use it either. It was therefore important for the customer that we were creative and tried to customize the design such that it looked like their sketches. This gives the user of the application the "feeling" and "branding" that the customer intended. We should also focus on the user experience, such as menus, buttons, and clicks and the response time should be fast.

The application should be developed with teachers and students in mind. This means that the system should be intuitive and easy to use for them. When it comes to the functionalities, it is important for the customer that the discussed features get implemented. It should be finished such that the customer has something to show the teachers, students, and principal in order to test the application at different schools.

The code should be well-documented and readable for PeopleUknow and other programmers, so that is it easy to understand the code and it will be easy for them to continue working on it. The customer stated also that they wanted this application to work on many devices as possible. But since we had a restriction on time, we decided that we should focus on both Android and iOS, and not desktop.

3.5. Related Applications

There are a lot of applications out there, which are focusing on learning social skills for children and students. They all have some similarities to the application we are developing.

Team Shake

Team Shake makes it possible to choose teams for board games, sporting events, tournaments, school projects or anytime groups are required [31]. This is done by entering all participants names in the application and give it a shake.

After School

After School is an anonymous message board app made for students [1]. Each school have their own message board that is only accessible to students of the given school. These boards are meant to encourage social interaction among students, sharing thoughts and uploading pictures. Teachers and parents can't access these boards and every student is anonymous.

Social Skill Builder

This application teaches children social thinking, language and behavior related to everyday situations [41]. It contains 19 modules with videos and questions for the user to see and react to by answering multiple choice questions. The videos in the application shows scenarios over real interactions in school.

Let's be Social: Social Skills Development

The application has written lessons and video lessons and it has the ability to create customized lessons from scratch [42]. It is designed to help special need professionals and parents teach social skills to those that struggle with social communication.

The Social Navigator

The Social Navigator is an application that assists children with social and behavioral challenges by helping them to adapt their behavior and developing life-long social skills[29]. The user can enter their current social situation and the application will instantly generate corresponding strategies and recommendations. This will guide them through how to handle the occurred situations.

Comparison of Different Application

We found several applications containing some of the functionality our application was supposed to contain, but none had all of it(see Table 3.1). Furthermore, most were dedicated either to teach and improve social skills, or to improve learning and well-being in the classroom. None of the applications focused on both, and in addition, many of the applications had little or no interaction between the users. From this research, we can conclude that our application have no directly competing applications already on the market, but a lot of applications that provide parts of the functionality.

Table 3.1, shows the similar applications and the different functions our customer wanted. The last column shows the users of the application. From the table, we can see that they provide some of the functions our customer wanted, but not all of them is covered.

	Team / Get-Toknow	Comments-and-Sharing	Exercise	Users
Team Shake	Add names, choose amount of team and then the team get assigned	Not provided	Not provided	Friends, class, groups
After School	Not provided	Students are able to share everything anonymously	Not provided	Students
Social Skill Builder	Provide games for team	Partly	Exercise with videos and questions with scenarios over real interactions in school.	Students, parents
Let's be Social	Not for teams	Not provided	Voice recording, video exercise and story with image	Children with special needs, parents and teachers
The Social Navigator	Not provided	Not provided	Prepare children for upcoming events by teaching social skills	Children with social challenges, doctors, teacher and parents

Table 3.1.: Application Functions

3.6. Choice of Solution

We are expecting to create a prototype with Scrapbook, GetToKnow, Exercises, Home, and Login. We will not create Login with Feide since PeopleUknow has not currently made an agreement with Feide. This would add complexity to the project and we feel the resources would be better spent elsewhere. The prototype we will make is part of the MVP. The different aspects of the solution will likely be just a beginning of the larger project and PeopleUknow does not expect a final product with all of the functionality that is described. We will continue development using React Native as programming framework.

The customer agreed with our assessment. We emphasized the that Feide login system would not be prioritized and that we would focus on the MVP.

4. Planning

In this chapter, we describe how we organized our project, documentation and management tools we used, and we describe the choice of lifecycle-model.

The last section of this chapter describes the risks for this project.

4.1. Organisational Demands

Our purpose for doing this project is to learn more about how to work with a real software development product, which involves planning, idea phase, implementing and testing. From the faculty and the course compendium there were some demands that the group needed to follow:

- The group needed to attend all the course lecture.
- If there was some women in the group, one of them needed to be the project manager.
- Every Friday the group needed to send a weekly report to the supervisor.
- The group needed to meet the supervisor every week.

4.2. Project Organization

Each member had been assigned to a role. The roles in this group were project leader, test leader, scrum master, security lead, quality assurance lead, system architect and customer interaction. Who had which role can be found in Table 4.1. The person with that specific role had the responsibility in that area, but all members were helping each other.

Figure 4.1.: the team of 6 members

Team roles	Name and E-mail	Study
Project Leader	June Kieu-van Thi Bui jkbui@stud.ntnu.no	Computer Science Software
System Architect	Ivar Helland Hessevik ivarhh@stud.ntnu.no	Computer Science Algorithms & HPC
Scrum Master, Security Lead	Christian Barth Roligheten chribrol@stud.ntnu.no	Computer Science Databases & Search
Quality Assurance Lead	Vigleik Lund vigleikl@stud.ntnu.no	Computer Science Databases & Search
Test leader	Matias Halsteinli Unsvåg matiashu@stud.ntnu.no	Computer Science Software
Customer Relationship Manager	Kristoffer Flinckenhagen jmfincke@stud.ntnu.no	Computer Science Databases & Search

Table 4.1.: Team roles and contact info

4.2.1. Role Descriptions

Under each roles get described. Figure 4.2 shows the overview of the different roles.

Project Leader

The project leader has an overview over the whole project, they are responsible for managing the resources and make sure there is no internal conflicts in the group. They are responsible for delegating tasks to other team-members, and making sure the tasks get done.

Customer Relationship Manager

Customer Relationship Manager (CRM) has the responsibility to bridge the communication between the group, the customer and the supervisor. They are responsible for official communication between the the project group and the customer. As well as sending weekly reports to the supervisor.

System Architect

The system architect is responsible for the architecture of our solution. They should make sure that any architectural choices are sound, and that they reflect customer demands.

Test Leader

The test leader is responsible for formalising and all types of testing performed on our solution. They are also responsible for coordinating testing with the customer when necessary.

Scrum Master

The scrum master is responsible for implementing and maintaining the scrum methodology within the group. They are also responsible for leading meetings related to the scrum process; such as holding the sprint meeting, sprint review and sprint retrospect. They are also responsible for maintaining the project and sprint backlog, and organizing the Kanban board.

Security Lead

The security lead is responsible for making sure the solution meets the criteria for security that we have defined. They are also responsible for making sure the security criteria we have defined are sound.

Quality Assurance Lead

The quality assurance lead is responsible for monitoring the software development, ensuring high code quality and to make sure that the product meets the specified requirements.

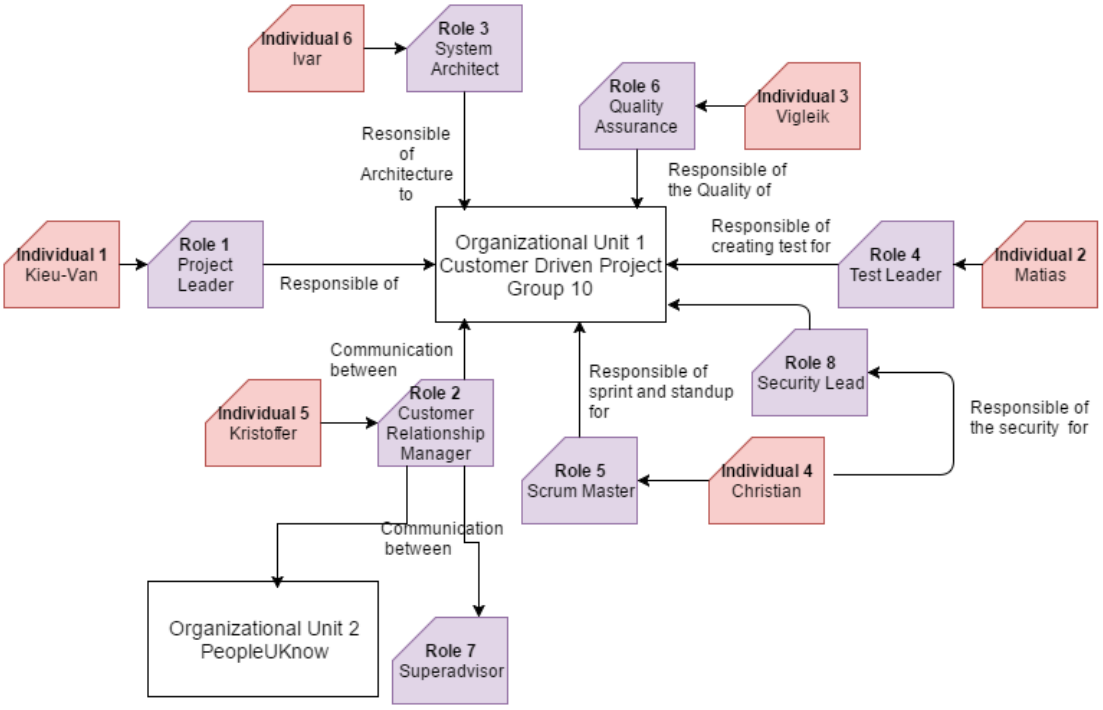


Figure 4.2.: Overview of the Roles

4.3. Quality Assurance

Quality assurance was an important part of the project because we were to deliver the final product to the customer and let them take over development; It is therefore important that we maintain good quality standard on our code as well as any documentation we deliver to the customer.

4.3.1. Quality of Code

In order to maintain a good coding standard we decided early on a appropriate code standard. The standard we chose was based on the standard used in examples on the documentation for the React Native framework.

4.3.2. Quality of Documentation

The standards for documentation depends on whether we expect the documentation to be used by developers or non-developers. Documentation relating to the code was written so that a developer with at least minimum understanding of software development could understand it.
Documentation written for non-developers was written so that anyone who knows how to use a computer could understand it. Any documentation written was also tested with the target persons so that we knew they understood how to make use of it.

4.3.3. Response Time

It is important for us to get response from the customer since the system is developed for the customer. If there was something we were uncertain about, it was therefore necessary to clarify this with the customer. To prevent delays in our development process, we agreed on the response time to be 24 hours, but when the customer was away due travelling the response time could be within 48 hours.

In Table 4.2 we have listed the time of response we used, which is from the compendium [20] provided by the course.

Description	**Time**
Approval of minutes of customer meeting	24 hours
Feedback on phase documents the customer would like to review	48 hours
Approval of phase documents	48 hours
Answer to a question	24 hours
To get agreed documents etc	24 hours

Table 4.2.: Time of Response

4.3.4. Document Review

For our routines for approval of phase documents, we made sure that group members were aware of the changes that were made. When the specific documents were updated, the group would review the changes.

We also sent some part of the report to the customer so they could see what we have done, if we understood each other, and if there was something the customer was missing.

4.3.5. Project Meetings

During our project there were a lot of meetings. Therefore it was also important to have templates for each meeting to save time and to use our time effectively, and to get most of it from each meeting.

Meeting with Development Team

Since members of our team has different schedules, some were unavailable during the week. We chose to have our weekly group meeting every Friday. The weekly meeting was used for preparation of meeting with the supervisor and other project related activities. These meeting was generally related to the management of the project. The agenda for the meetings can be found in B.1.1.

Daily stand-ups Meetings

An important part of the Scrum-methodology we decided to use (see Section: 4.4.4) is daily stand-ups. Each stand-up should be about 15 minutes and it is discussed what each member has done since the last time. Since members of the group have different schedules, that varies from week to week, stand-ups were not strictly daily. The meeting should be held each morning and discuss plans for the day and what the members did the previous day. Other smaller issues regarding development should be discussed after the stand-up.

Customer Meetings

We scheduled to meet with the customer at the beginning of every sprint. An alternative would be to have one meeting at the end of each sprint, and one at the beginning. To save time we merged these meetings, and talked about both the previous sprint, and the upcoming sprint in the same meeting. It was the customer relationship manager, which had the responsibility to schedule this. The customer meetings were used to discuss progress and decide tasks for the sprint. These meetings could also be used to show prototypes and get feedback for the next sprint cycle. The duration of the customer meeting should be about one hour. The template used for these meetings can be found in B.1.3.

Supervisor Meetings

Supervisor meeting were held every week. The comments from the weekly group meeting were reviewed and we were given feedback on the progress of the report. Agenda for meeting can be found in B.1.4.

Weekly Report

Every Friday, the report was sent to the supervisor before 12.00. This was to ensure that our supervisor got time to have a look at our report in order to give us feedback and comments on the document. The weekly report followed a template (See B.1.5). The format of the weekly report contained what had been done, what we should do, any questions and the report.

4.4. Choice of Lifecycle-model

This section presents possible choices for development methodologies in software development and which one we used for our project.

4.4.1. Waterfall

Waterfall is a methodology for software development. This method is a linear approach. The sequence of this approach consist of five phases, each of which are finished sequentially. For example, when the requirements phase is done, no more requirements are added to the project (See figure 4.3).

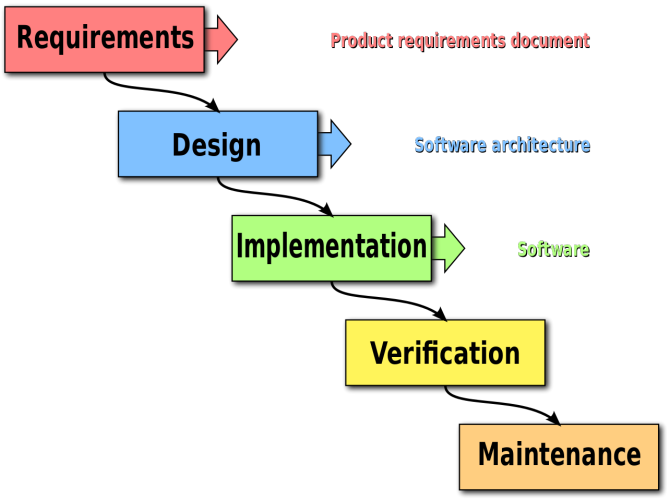


Figure 4.3.: Waterfall method [47]

.

4.4.2. Agile

Agile development describes a set of principles for software development [20]. It allows the requirements and solutions to evolve through collaborative effort of self-organizing cross-functional teams. By having agile approaches, it can help teams to respond to unpredictability through incremental, iterative work and empirical feedback.

Scrum

Scrum is an agile methodology that focuses on incremental changes. [21] The methodology introduces feedback-loops to encourage developers to inspect and adapt the product. A product owner creates a prioritized list (product backlog) for the product. During the sprint planning, the project team takes elements from the product backlog and places them in the sprint backlog. The team has a time-frame for each sprint. Daily Scrum meetings should be give progress-reports to the team, and the scrum master keeps the team focused. The work should be finish in the end of the sprint. The team does a sprint review and sprint retrospective on the process. This concludes a sprint and the team can begin a new sprint. The process is documented in the burndown chart.

XP

Extreme Programming (XP) is an extreme case of agile development [20]. XP focuses on flexibility. Code, testing, listening and designing are the core features of XP. Pair programming and test driven development are important in XP. The communication within the team is often done as stand-up meetings.

Kanban

Kanban is an agile methodology like Scrum, but less structured [20]. Kanban focuses on visualization of the work flow and organizes the work on Kanban boards. These boards consist of cards placed in different columns. The columns represent the different development processes while the cards represent the work-tasks. The cards move from left to right along the columns as they pass the different development stages. Each column is limited to a pre-defined number of cards. The limits are crucial for avoiding overproduction, revealing bottlenecks and optimizing work flow.

4.4.3. Comparison of Methodologies

In table 4.3, we compare the different pros and cons with the different methodologies.

Methodology	Pros	Cons
Waterfall	Structured, simple, easy testing	Difficult to make changes, not fit for long projects
Scrum	Incremental, suitable for small projects	Not suitable for big project
Extreme Programming (XP)	Customer in center, Quick iterations	Frequent meeting creates overhead,
Kanban	Graphical view of tasks, flexibility	Boards must be updated, board can be overcomplicated

Table 4.3.: Comparison of different software methodologies.

First we had to choose whether to use some agile method or the more traditional waterfall method. Whilst the waterfall method is suitable for projects that are well-specified, this was not the case for our project: The customer, being a startup, had a general idea of the features to be included in the application, but had little detailed knowledge of how these could best be implemented. A lot of new features the customer wanted were also discovered during workshops and meetings between the customer and project team. With this in mind we chose to use an agile method, as these are better suited for changing requirements and specifications.

4.4.4. Selecting an Agile Method

For our project we mostly considered XP and Scrum as candidate agile methods. We found that whilst XP have many good qualities we want in the project, we had too little time and experience to use it effectively on such a short project. Scrum then came out as the winner because the idea of sprints were well suited for the tasks we needed to work on for the prototype. Many of the team members also had past experience with Scrum. We chose to make use of a Kanban board to organize our work, this was to get a better overview of the different tasks, and what each team-member was working on.

4.5. Phases

We decided to split our project into seven phases: introduction, preliminary study and planning, four sprints, and final phase. For each phase we planned a general set of goals that we should complete. These goals were as high-level as possible, since details about how these should be implemented could easily change throughout the project. The final phase was used for preparing the report for delivery and preparation for the final presentation.

Introduction phase was the first phase of our project. This phase was where we got to know each other on the team and the customer. Also, we use this time to get a clear understanding on what this course was really about and reading the compendium.

During the preliminary and planning phase, we worked closely with the customer to nail down the scope of the project; we defined requirements and priorities, decided the architecture to use and figured out how we would work as a team. We also did the pre-study during this phase.

Following the preliminary and planning phase was four sprint phases; each sprint phase we had agreed on a set of features to implement and focused most of our effort on implementing these. User-testing was also performed throughout these phases. User-testing is discussed in more detail in chapter 10. In the final phase, we focused on finishing the report, fixing small things in the application and preparing for the presentation.

The table 4.4 shows summaries of the different phases and describes their duration.

Phase	**Description**	**Week**
Introduction	Getting to know the course, the team, and the costumer	34
Preliminary and Planning	Planning, setup of project, and research	35-36
Sprint 1	Learn React-native, create prototypes, implement login	37-38
Sprint 2	Security, user-testing, implement Exercise and Scrapbook	39-40
Sprint 3	Implementation of GetToKnow and home page	41-42
Sprint 4	Usability testing, fixes and report	43-44
Final phase	Report writing and presentation	45-46

Table 4.4.: Project Phases

4.6. Work Breakdown Structure

By using Work Breakdown Structure it helped us to allocate the project resources by breaking the project into smaller components. We could therefore track and identify which parts of the project require most time and identify issues and problem areas in the project organization.

- **Research**: contains all the tasks we did related to research such as background information from PeopleUknow, research on which technologies to use and similar application etc. I also included evaluating and learning how to use the different technologies.
- **Meetings and planning**: includes all the meeting activities we had with our customer, supervisor, internal in the group and with the customer's technical advisor. Planning goes under all the planning before each sprints.
- **Design**: consists of all the tasks for prototyping and for how the application should look like.

- **Implementation**: includes all of the implementation of the application and the testing of the application.
- **Documentation**: is all the activities we did for writing the report and documentation of the application which can be found in the appendix.

The following work breakdown structure shows the parts of the project we determined were most important regarding allocating resources and time during the project. This allowed us to easier see how much time would need to be spent on the different parts of the project to get the best result.

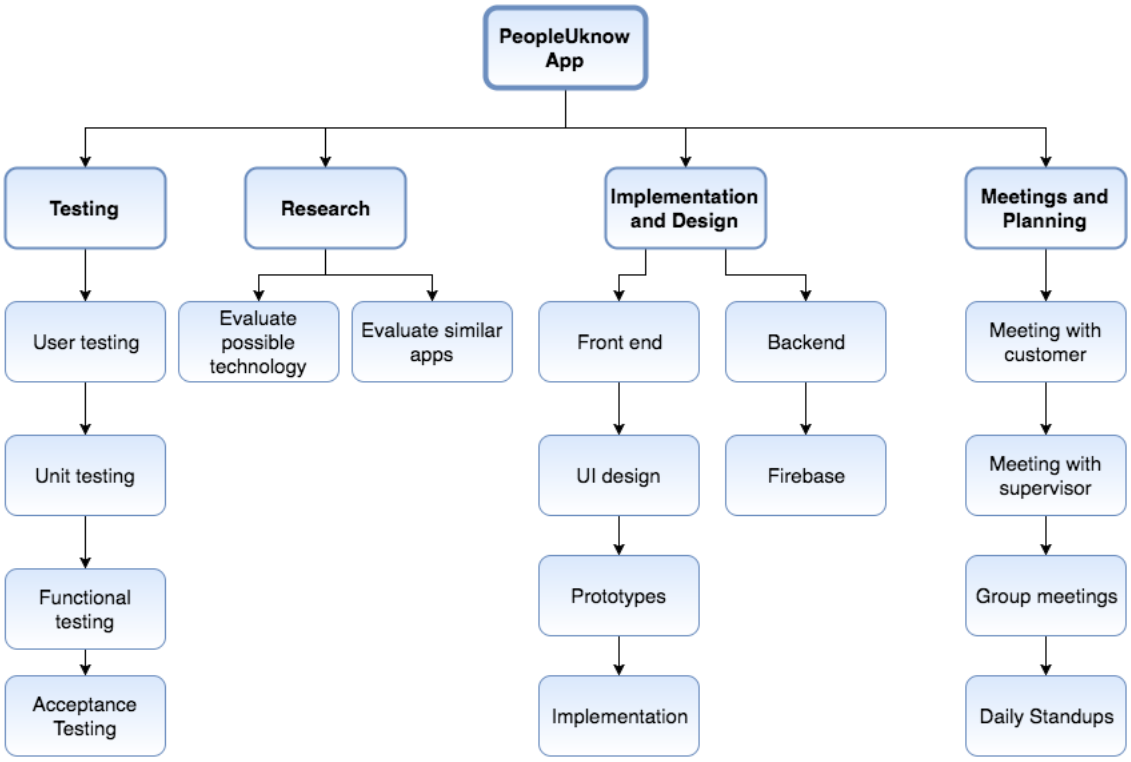


Figure 4.4.: WBS diagram showing the different parts of the project

4.7. Gantt

The overall plan for this project is represented as a Gantt Diagram in Figure 4.5. The Gantt chart can be used to see how we planned to schedule the development of the project, from start to end. It also gives a summary of the project work; for instance in the beginning of each sprint the group had a sprint meeting.

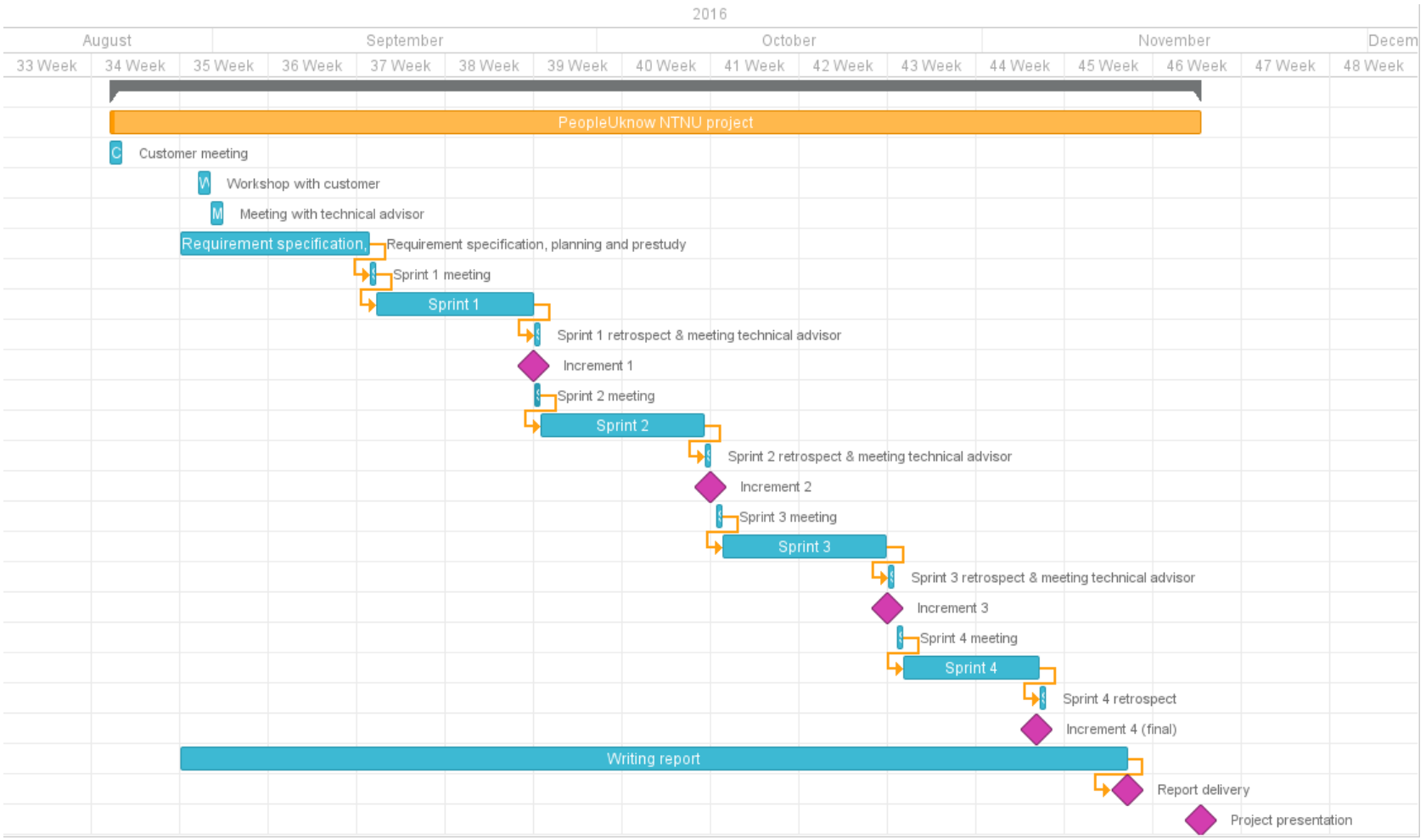


Figure 4.5.: Gantt diagram showing the the project plan with phases, important meetings and milestones

4.8. Milestones

In the table 4.5 we have listed specific points along the project development time-line. These milestones help us to get progress by trying to reach them at the given time.

Milestones	**Date finished**
Pre-study	09.09.2016
Sprint 1 ended	23.09.2016
Sprint 2 ended	07.10.2016
Formative Usability testing	25.09.2016
Sprint 3 ended	21.10.2016
Summativ Usability testing	26.10.2016
Sprint 4 ended	04.11.2016

Table 4.5.: Milestones

4.9. Risk Management

In this section we describe how we did the risk management, how we found the risk in the table 4.6 (which is ordered by the risk) and how we prioritize the risk (see table 4.8).

Risk Factor	**Risk**
Misunderstanding between the group and the customer	Very H
The customer unavailable	Very H
Underestimaded the time to complete the tasks	H
Unforeseen technical issues	H
Lack of knowledge	H
Conflicts in the group	H
Members of the group get sick, or are not showing up	M
Loss of data on Google Drive or Sharelatex	M
Communication problems	M
Tool risk	M
Wrong choices have/has been made in the middle of process	M
A project member drops the course	M
Low motivation	M
Things/materials get stolen	L
Unable to find work place	L

Table 4.6.: Founded Risk

4.9.1. Identify Potential Risks

The first thing we did was to identify risks. This is necessary to understand what can possibly go wrong during this project and to create a common understanding of the

risks. The identifying of risks was conducted by listing all the possible things that could happen to our project, while we discussed it with each other.

We were also looking at the cause of the problem to get a better understanding, and who had responsibility for this risks. This made it clearer to see how big impact the project has on the risk because if we cannot find the cause of the problem, it may be that we do not mitigate the actual problem.

4.9.2. Determine Likelihood and Impact

To determine the likelihood and impact of the risk, we follow a template provided by Difi.no[6]. In this report, we describe the **Likelihood** as the probability of the risk occurs (which is from 1 to 10). Where the likelihood between 1-3 is low, likelihood between 4-7 is medium and likelihood between 8-10 is high. **Impact** is defined as the consequence of the risk from 1 to 10. Where 1-3 describes the impact is low, 4-7 is medium and 8-9 is high for the project goal, deadline,and framework.

When we multiply the likelihood and the impact, we get the risk (*Risk = Likelihood x Impact*). Figure 4.7 shows the likelihood and the impact, which gives the risk. The red areas on the figure is where the risk are high and the green is where the risk is low.

Impact / Likelihood	Low	Medium	High
High	M	H	H
Medium	L	M	H
Low	L	L	M

Table 4.7.: Risk Table

4.9.3. Mitigation, Implementing and Control

For planning to mitigate the risks, we discussed the "strategy and action" on how we were going to avoid, reduce, transfer or accept the risks. More on the mitigation on different risk can be found in Section 4.9.4.

4.9.4. Risk Table

In this subsection, we have created risk tables for all the risks that were identified for the project which can potentially occur. Each risk has been assigned to a risk ID. In the tables, the activity row describes which of the activities in the project are affected by that risk. We use H (High), M (Medium) or L (Low) for describing the impact and the likelihood (see Section 4.9.2). Strategy and action row is the mitigation part, how we want to avoid, reduce accept etc. While the last row, responsibility describes the person that have the responsibility for the risk if it occurs.

In the end we had to prioritize the risk according to their risk. For this we made a table 4.8 which contains the risk ID, risk factor, impact (I), likelihood (L), the risk

and the priority. The priority is from 1-3, where 1 is very important and 3 is not so important.

Risk Factor	**I**	**L**	**Risk**	**Priority**
Human Risk				
The customer unavailable	H	H	Very H	1
Misunderstanding between the group and the customer	H	H	Very H	1
Conflicts in the group	M	H	H	1
Communication problems	M	M	M	2
A project member drops the course	H	L	M	2
Low motivation	M	M	M	2
Members of the group get sick, or are not showing up	M	M	M	2
Process Content Risk				
Lack of knowledge	H	M	H	1
Wrong choices have/has been made in the middle of process	M	M	M	2
Loss of data on Google Drive or Sharelatex	H	L	M	2
Things/materials get stolen	M	L	L	3
Unable to find work place	L	M	L	3
Technical Risk				
Unforeseen technical issues	H	M	H	1
Tool risk	M	M	M	2
Estimation Risk				
Underestimaded the time to complete the tasks	H	M	H	1

Table 4.8.: Risk Table

5. Requirements Analysis

This chapter describes how we identified the requirements; both functional and non-functional requirements.

The functional requirements are represented textually and as use case diagrams. The functional requirements are described with user stories. The non-functional requirements will be described in a context of usability, modifiability, security and portability. We will also describe our approach for estimating the time for each use cases.

5.1. Requirement Elicitation

At the start of the project the customer had a lot of features they wanted to implement, but because we had a total of 12 weeks to work on the application we had to constrain ourselves to some of the core features the customer wanted. To get an overview of all the functional and non-functional requirements, we held a workshop with the customer where we discussed the different features they wanted and agreed on what was most important.

We had an internal meeting after the workshop, where we discussed what the customer wanted. From this discussion we managed to create a list of functional requirements and a corresponding list of non-functional requirements, which can be found in Section 5.2 and 5.4. To further confirm that the customer was happy with the progress, we reaffirmed what was important at each sprint meeting. We also agreed to implement a Lunch Date feature if we had extra time. The reason for this extra feature was that we did not know how quickly we could implement the core features.

5.2. Functional Requirements

At the end of the planning phase we had defined a set of functional requirements that we agreed we should focus on. The requirements we found in Table 5.1 are represented as user stories: as a user I want "to do something". Each stories have story points and a priority. The Story points are an arbitrary measure, which is used for measuring the effort to implement the story. The priority is from low to high, where H is the stories that we are prioritizing highest before Medium and Low. The method we used estimate the story points is described in Section 5.5.

The user stories have also been categorized into: Home, GetToKnow, Exercises, Lunch date, Login and Scrapbook. The Figure 5.1 shows the categorization of the user stories.

ID	User-story	Story points	Priority
F1	As a user I want to be able to log in to my account with email and password.	58	H
F2	As a user I want to be notified if I am not successfully logged in.	2	M
F3	As a user I want to view pictures and text other users in my class has made to the class scrapbook, and see custom styling chosen by classmates on posts.	40	H
F4	As a user I want to be able to comment on posts made by classmates to the scrapbook.	10	L
F5	As a user I want to be able to post pictures and text to the class scrapbook, and choose the styling for the post.	20	H
F6	As a student I want to view which pairs me and my classmates have been assigned to by the teacher.	20	M
F7	As a teacher I want to group students into pairs, I want to be able to select who is attending and modify the pairs before my students can see the pairings.	70	H
F8	As a teacher I want to view all exercises I can choose, and select those I want student pairs to do.	25	H
F9	As a student I want to be able to view exercises picked for us by the teacher.	10	M
F10	As a user I want to be able to search the scrapbook on text and other post-related data.	10	L
F11	As a student I want the ability to sign up for a lunch-date with another student.	20	L
F12	As a student I want the app to pair me with another student for a lunchdate, I want to be notified when this happens and be able to view who I was paired with.	20	L
F13	As a user I want to easily get an overview of the app features, and be able to open the page for the feature I want to use.	20	H

Table 5.1.: User Stories

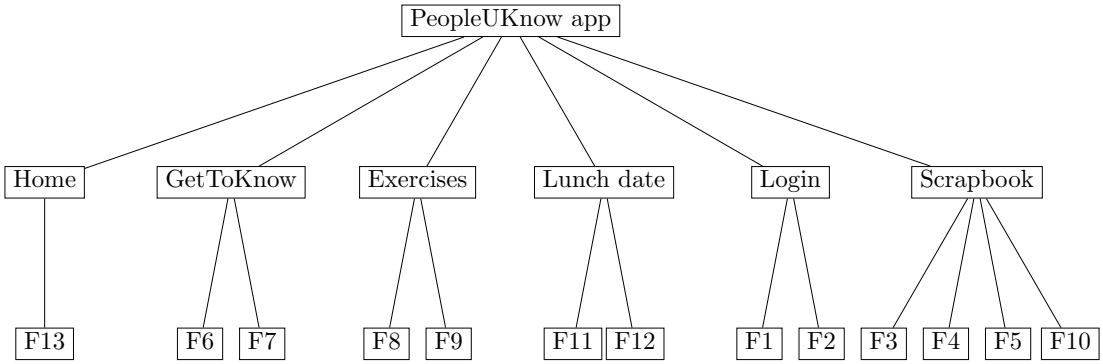


Figure 5.1.: Categorized User Stories

5.3. Use Case

In this section we will be presenting the use cases from the functional requirements. Use case is used for analyzing, identifying, clarifying and organizing the system requirements [35]. Based on the requirements from Table 5.4.1 we have created user stories, textual use cases and use case diagrams . The important use cases have been listed in this section while the rest of them can be found in the Appendix A.3.

Use Case #	1
Application	PeopleUKnow App
Name	Login
Reqquirements	F1, F2
Description	A user wants to log in to the PeopleUKnow application with his or her email and password.
Primary Actors	Students, teachers and parents
Preconditions	User has an account
Trigger	
Basic flow	• The user is presented with login page. • The user types in email and password. • The user push the login button or press enter. • The system confirms that the user is logged in.
Alternative	
Post conditions	The user is logged in.

Table 5.2.: Use Case 1

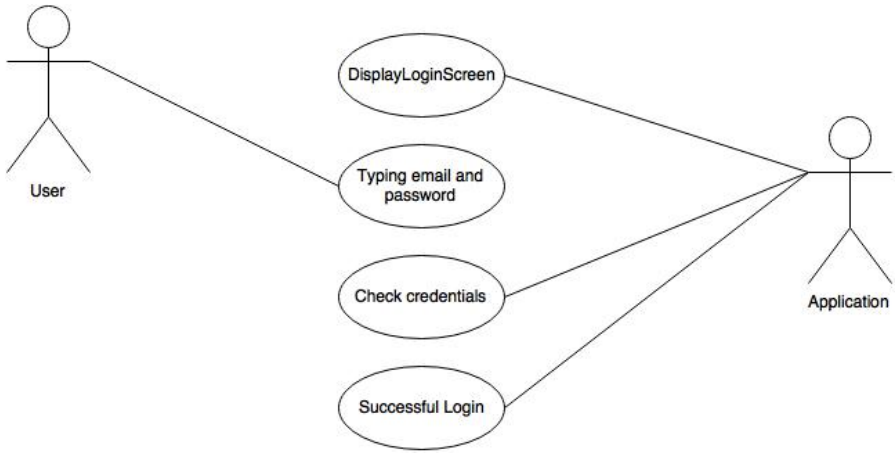


Figure 5.2.: Use Case Diagram 1: Login

Use Case #	4
Application	PeopleUKnow App
Name	Sharing on the Scrapbook
Requirements	F5
Description	The user wants to share a picture and text on the scrapbook and choose the style for the post.
Primary Actors	Students
Preconditions	• The user is logged in • The user is viewing the scrapbook
Trigger	The user has finished an exercise and wants to share it.
Basic flow	• The user presses the share button • The application shows a form for uploading posts • The user chooses a different styling • The user writes a description of the post • The user chooses a picture • The user chooses one or more tags to describe the post • The application uploads the post • The user can see the post
Alternative	
Post conditions	The user has shared a post with the preferred style.

Table 5.3.: Use case 4

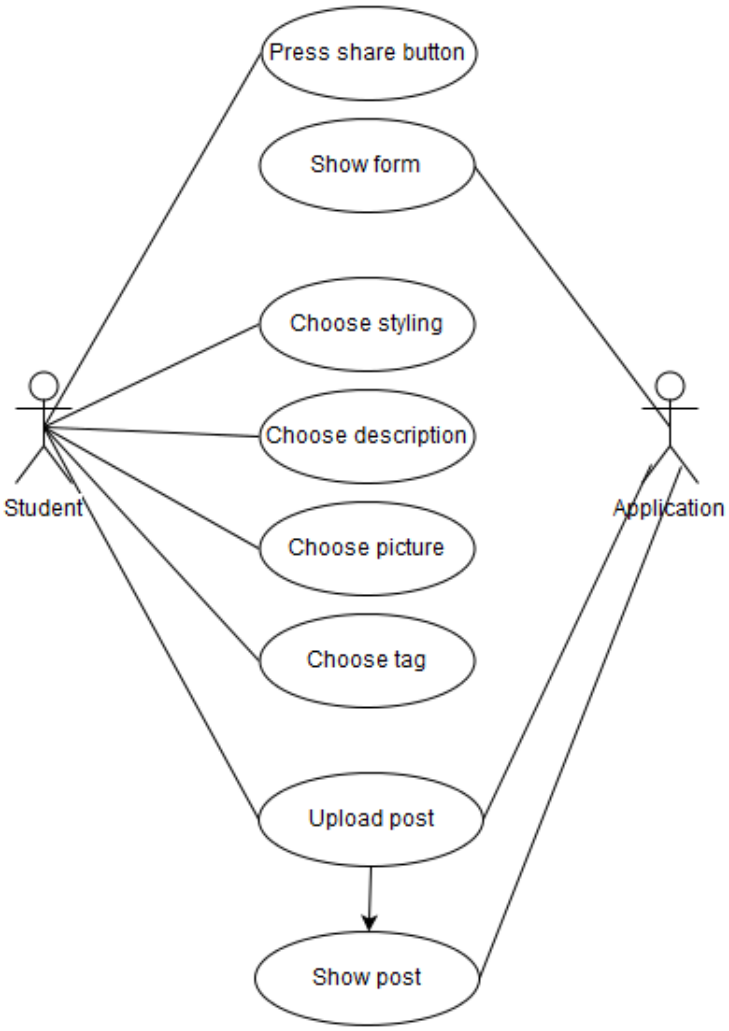


Figure 5.3.: Use Case Diagram 4: Sharing on the Scrapbook

Use Case #	6
Application	PeopleUKnow App
Name	teacher Groups Students into Pairs
Requirements	F7
Description	A teacher wants to group his/her students into pairs, and modifying the pairs if is necessary.
Primary Actors	Teachers
Preconditions	• The user has an account and is logged in • The user has a class
Trigger	The teacher has some exercises for the students, and therefore he/she need to group the students.
Basic flow	• The system shows the menu bar • The user choose “GetToKnow” function • The user push the button pair teams • The user can see all the pairs created • The user edit some of the pairs • The user can see the pairs created after modifying them
Alternative	
Post conditions	The user has created and modified teams

Table 5.4.: Use case 6

Figure 5.4.: Use Case Diagram 6: View the GetToKnow pairs

Use Case #	8
Application	PeopleUKnow App
Name	Student View the Chosen Exercises
Requirements	F9
Description	The user wants to see what exercises have been chosen by the teacher.
Primary Actors	Students
Preconditions	• The user is logged in. • The teacher has chosen exercises.
Trigger	
Basic flow	• The user choose “Exercise” button from the menu bar • The user views the exercises
Alternative	
Post conditions	The user views the exercises

Table 5.5.: Use case 8

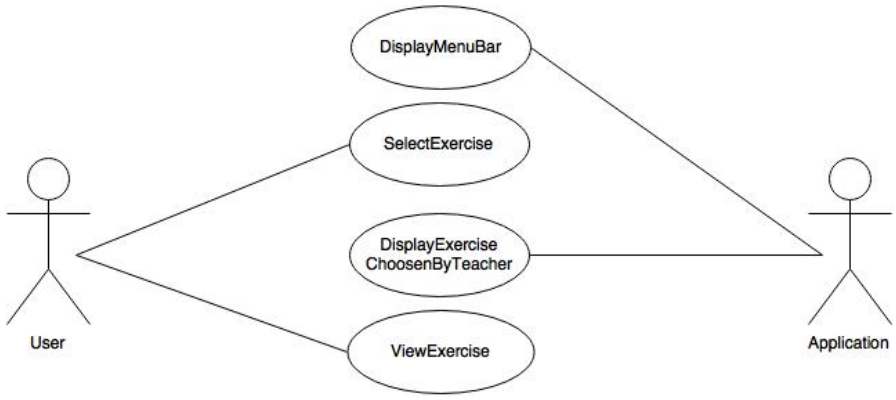


Figure 5.5.: Use Case Diagram 8: Student View the Chosen Exercises

5.4. Non-functional Requirements

This section describes the non-functional requirements, which is how the system should implement the functional requirements [38].

5.4.1. Quality Attribute Requirements

For the application to be successful, it needs several quality attributes. In Table 5.4.1 we have listed the requirements we feel are most important for this application. They are grouped related to their quality attributes:

- U = Usability
- M = Modifiability
- S = Security
- P = Portability

ID	Description	Rationale	Priority
U1	The user must be able to learn how to use a new feature, e.g. Exercises, within 5 minutes of learning.	The app must be intuitive and easy to use. If the users find the app cumbersome to use and difficult to understand they will use it less, or not at all	Essential
U2	The user should not need to remember previous actions or information in the app to use functionality.	If the users needs to remember a lot of details from previous screens they can get fatigued by using the app, and may fail to use some of functionality properly	High
U3	All buttons should perform an action or give feedback to the user	Feedback is essential for user experience. Users tend to be confused by buttons without feedback. They can be unsure if they actually pressed it, or it can be hard to understand what it does.	High
U4	All features should be designed to be intuitive enough to use without instructions, if this cannot be achieved, functionality that teaches the user how to use it should be provided.	How to use certain features may be hard to understand when they use the application for the first time. Instead of just trying everything until they understand it, some sort of instructions should be provided.	Medium
M1	It should be possible to implement new features without changing the existing features.	Since PeopleUknow wants to continue building on the app, the ability to easily create more functionality is very important	High
M2	The login feature should be replaceable with Feide-login without major refactoring	The app is intended to be used in school, and PeopleUknow want the possibility for the app to be connected with Feide in the future	Medium
P1	The application must work with Android phones and iOS phones that are younger than 5 years.	If students in a class are supposed to use the app, it needs to work on most phones, since teachers cannot expect students to buy new phones to get the app.	High

Table 5.6.: Non-functional requirements

5.4.2. Quality Attribute Scenarios

In the following tables we describe the non-functional requirements in context of Quality Attribute Scenarios. These can be used later by PeopleUknow to evaluate whether the

system meets the non functional requirements.

Scenario - U1	
Source of stimulus	User.
Stimulus	Learn to use a feature in the application.
Artifact	Application.
Environment	Application at runtime.
Response	User uses the feature.
Response measure	Is able to find and use the functionality within 5 minutes of learning.

Table 5.7.: Scenario - U1: User learning

Scenario - U2	
Source of stimulus	User.
Stimulus	Uses functionality and navigates in the app.
Artifact	Application.
Environment	Application at runtime.
Response	User is able to navigate to different features and use the desired functionality in the app.
Response measure	Is able to find and use the functionality without asking for help, or going back to check previous stages.

Table 5.8.: Scenario - U2: Intuitive usage.

Scenario - U3	
Source of stimulus	User.
Stimulus	Uses functionality and navigates in the app.
Artifact	Application.
Environment	Application at runtime.
Response	User is able to navigate to different features and use the desired functionality in the app.
Response measure	Is able to find and use the functionality without asking for help, or going back to check previous stages.

Table 5.9.: Scenario - U3: Feedback from interface.

Scenario - U4	
Source of stimulus	User.
Stimulus	Uses tries new functionality.
Artifact	Application.
Environment	Application at runtime.
Response	User is able to use the new functionality.
Response measure	User is able to use the functionality on his/her own, or by using help-button or something similar to understand it, within 5 minutes of learning.

Table 5.10.: Scenario - U4: Help when needed.

Scenario - M1	
Source of stimulus	Developer.
Stimulus	Tries to create a new component.
Artifact	Application.
Environment	Application at design-time.
Response	Developer adds new functionality.
Response measure	New component can be added without creating problems for the existing components.

Table 5.11.: Scenario - M1: modifiability of code.

Scenario - M2	
Source of stimulus	Developer.
Stimulus	Tries to change the login-functionality to implement Feide.
Artifact	Application.
Environment	Application at design-time.
Response	Login feature is replaced.
Response measure	The login with email is replaced by Feide-login, without affecting other features.

Table 5.12.: Scenario - M2: Possibility for integration with Feide.

Scenario - S1	
Source of stimulus	User, student.
Stimulus	Tries to change the parings in the GetToKnow feature.
Artifact	Application.
Environment	Application at runtime.
Response	Access is denied.
Response measure	User is unable to access the functionality.

Table 5.13.: Scenario - S1: Verification of data.

Scenario - S2	
Source of stimulus	User, student.
Stimulus	Tries to see images from the scrapbook of another class.
Artifact	Application.
Environment	Application at runtime.
Response	Access is denied.
Response measure	User only able to see posts from their own class.

Table 5.14.: Scenario - S2: Access control.

Scenario - P1	
Source of stimulus	User.
Stimulus	Tries to install and use the application.
Artifact	Application.
Environment	Application at build-time, runtime.
Response	Installing and launching is successful.
Response measure	Less than 10% of users are unable to install or use the application due to Operating System specifications.

Table 5.15.: Scenario - P1: Compatibility with devices.

5.5. Estimation of Realization Effort for Use-Case model

In order to estimate the effort of developing the user stories, we played planning poker[48] which can be seen in Figure 5.6. Each team member got a set of cards with numbers. Then instead of saying the estimation value out loud, each player played with the numbered cards face-down to the table. By using cards it avoid the cognitive bias, since the first number that is spoken will set a precedent for the estimation of the user story. The values should be approximately equal. In the case where the numbers of the cards are very different, the group should discuss it further. When the values are approximately equal, the estimation is an average of the card values.

Figure 5.6.: The Team Playing Planning Poker.

The estimation can be seen in Table 5.1,where the estimations are the story points. Story points are arbitrary measure, this tells the team how hard each story is.

6. Quality Assurance

In this chapter we describe the programming environment, coding standard and code review procedure.

6.1. Programming Language

As previously mentioned, the project uses the React Native framework. This is a JavaScript framework. There is no programming required to use the backend side of Firebase. Further details and rational behind the decisions are described in Section 7.4.

6.2. Programming Environment

While there does exist IDEs for React Native, the members of team were free to choose the programming environment they were most comfortable with. Popular among the group was Atom [3] and Sublime [43] with some addons. Facebook uses an extension of Atom which is called Nuclide [9]. It is designed for React Native development and has useful features.

6.3. Coding standard

Since the customer was probably going to improve upon our code after our project was finished, it was important that the coding style we chose was consistent with customers expectation and our individual expectations inside the team. In order to facilitate good coding practices we decided early on common set of rules that all developers would follow to ensure consistent style throughout the project.

Indentation All developers had to indent using only spaces. Each level of indentation increase with two spaces.

Comments Code blocks with high complexity should be commented with an explanation of the actions taken in that block.

ESDoc tags All classes and methods made by developers had to be tagged with the appropriate ESDoc tags: Methods had to have the @param and @returns tags when appropriate. Classes should have a high-level description of its purpose. Overridden methods from the React Native framework were except from this rule and required no ESDoc tags.

Variable and class naming convention All variable and class names should be written in camel case. Variable and class names should describe their purpose.

6.4. Code review

In order to make sure the coding standard described in Section 6.3 was used, all changes to the main branch on the Github repository had to go through code review. Code review consisted of one developer who was not involved in the changes looking through the code to make sure it met the coding standards.

The code also had to be tested to make sure it worked. Testing was done by compiling the app and manually trying out the features changed or introduced. All changes that made use of features specific to iOS or Android had to be tested on both an Android and iOS device.

7. Technologies

In this chapter, we have described the different technologies and tools that we used for the development of the project.

7.1. Version Control

We used the Github[16] service as version control system for our code and documentation. Github allows users to create private repositories that make use of the well-known version-control system Git to create a full history of every change that has happened to the code over time. We chose Github mostly because the customer was already using the service to host what code they had before we started working on the project. However we would have probably used Github if the choice was left to us; Github is very easy to use and most of our team was already familiar with the service beforehand.

The version control procedure was handled with pull-requests. A pull-request allows a developer to tell others about changes pushed to the repository. We organized features in branches and when the feature was finished, we created a pull request. This allowed us to review each others code and manage changes before they were accepted into the application.

7.2. Management and Communication Tools

To manage the project and the communication in the group we used different tools such as Google Drive, Trello, Facebook, and Slack. All of these tools will be described in this section.

7.2.1. Google Drive

Google Drive[18] was used in order to manage and store our materials such as figures, documents, and other files. The file management system provides 15 GB cloud storage for each user. Google Drive allows sharing of files and it could be shared with other users such as our customer. It is also widely used, and everyone in the development group had used it so it came naturally to use Google Drive.

7.2.2. Trello

Trello[44] was used to manage our Kanban board. Trello is an online project management tool that offers a simple way to share and organize projects into Kanban-style boards. People can register when they have started on a task, supply comments and an event history is shared to the group. Using Trello makes it easy for the development team to have an overview of the development-process.

7.2.3. Facebook

Facebook[12] was used for communication between members of the group and between our customer. We had 2 private groups on Facebook. The first group was for our group where we were communicated internally in the team. While the second group was for communicating with the customer. Facebook was used to keep track of events with the customer, and for asking any quick, informal questions.

7.2.4. Slack

Slack[39] was also used for communication. This tool made it possible for the project team to create channels to organize the conversations within the group. Members were also able to send direct messages to each other, which was private and secure. Slack makes it possible for the users to see the important messages or information such that they are not drowned by unimportant messages like on Facebook.

7.3. Documentation Tools

7.3.1. ShareLatex

In order to write the report, Sharelatex[37] was used. Sharelatex is an online LateX editor. It allows the group to collaborate in real-time and it requires no installation. All we needed was an account.

7.3.2. Google Docs

Google Docs[18] is web-based word processor. It allows for sharing of documents and simultaneous editing. Internal documents such as summaries of meeting, planning, sketches and various intermediate documents can easily be shared with the rest of the group. It will allow for easily sharing documents with the customer.

7.3.3. ESDoc

The code was documented using ESDoc[8]. ESDoc is a tool for automatic generation of documentation from documentation-tags for classes and methods in the code. The documentation-tags can be read as-is in the code or the ESDoc tool can generate a series of HTML documents that allow for easy viewing and navigation of documentation for classes and methods.

7.4. Frameworks and Development Tools

7.4.1. Firebase

Firebase is a mobile platform containing a suite of tools useful when building a mobile app[17]. Specifically Firebase allows developers to set up a backend for their apps without the hassle of writing a lot of code manually. Most features provided by Firebase are ready to use out of the box and can be configured for individual app needs.

Authentication Firebase supports user authentication from a wide variaty of sources; Developers can allow users to sign in to their apps using accounts from Google, Facebook, Twitter and others. The developers can also configure anonymous sign-in, where users create accounts specific to the Firebase project using email and password.

Real-time database The real-time database included in Firebase allows developers to create a cloud-hosted NoSQL database that is suitable for apps without advanced backend requirements. The database has no pre-defined structure and it is therefore easy to add new data to it. When using the database with the official API it is possible for apps to listen for database updates, this allows developers to create a highly interactive app that automatically updates itself on database events.

The database can be configured with rules that allow developers to restrict the type of data uploaded by the user. It is also possible to configure security rules that restrict who can read and write certain parts of the database. These rules integrate with the authentication feature allowing developers to restrict database access to certain users or user-groups.

Storage Firebase includes built-in support for Google Cloud storage. This allows developers to easily support download and upload of pictures, videos and documents in their apps. It also integrates with security rules in the real-time database, meaning developers can restrict access to certain files or directories on the cloud storage based on authentication and/or database values.

7.4.2. React Native

React Native is a framework developed by Facebook[11]. It facilitates cross-platform mobile app development by allowing cross-platform JavaScript code to communicate with native code for iOS and Android. This allows developers to easily access platform specific features such as the camera or the file system.

React components are written in an XML-like JavaScript extension called JSX. JSX allows developers to use markup-like syntax for defining tree structures with attributes and is supposed to help with readability and work efficiency. Since React transforms the JSX code into JavaScript, developers may also write the components in plain JavaScript.

React Native does not require the user to recompile the app to see changes. During development, the app is run on a Node.js server on the development machine and changes can be updated in real-time in the app. The programming framework also requires less code than an app written in Java and/or Swift. The disadvantage of using React Native is that the project is relatively new. The first public release of React Native was early 2016 and at the start of development version 0.28 of React Native was used. Thus it is a high chance that there are bugs in the framework itself. Furthermore, since it is a new framework, there is a smaller community of app-developers using React Native compared to regular Android/iOS development.

7.4.3. Drawio

Draw.io provides free tools for online diagram software for making flowcharts, process diagrams, org charts, UML, ER and network diagrams[7]. We used this tool to create most of our diagrams such as our use case diagrams, process diagrams etc.

8. System Architecture

In this chapter we describe the design of the system, and the architecture we implemented.

We have listed the most important Architectural drivers and patterns, as well as architectural views. The views are based on the 4+1 view model, by Philippe Kruchten[23]. The views represent the same system, but from different view points to give a thorough overview of the system.

8.1. Architectural Drivers

The larger software development projects are, the more the final result will be affected by the system architecture. To make sure we chose the most appropriate architecture for our system, we started by identifying the the elements that had greatest impact on the architectural choices we made: the architectural drivers.

8.1.1. Quality Requirements

Maintainability and modifiability

Maintainability and modifiability was very important to the customer since they planned to use the prototype we built as the base for a larger system. Since they would then need to modify or add features it was important that the code we produced was organized so that other developers easily could modify or add to our work.

When deciding on the architecture we also had to consider customer preferences: The customer told us early that they had experience with React Native and Firebase, and would prefer if we used these frameworks in our prototype.

Scalability

Even though the prototype we produced did not need to scale, the architecture we chose had to have the ability to scale in case the prototype we produced eventually became a real product. If we did not focus on scalability the customer might have had to make major changes to the architecture before releasing it to the world.

8.1.2. Functional Constraints

Privacy of users

It was important that any information users submitted to the system was kept confidential, it was also important that users were prevented from accessing information they were not supposed to access.

Multi-platform

The customer wanted the prototype to work on both iOS and Android phones. It was therefore necessary that the architecture supported integration with both these platforms.

8.1.3. Business Constraints

Time to market

Due to the short time-frame for the system, avoiding a very complex architecture, and choosing frameworks that were relatively simple to learn and use was ideal.

No budget

Since the project had no budget, it was necessary to avoid frameworks, tools or services that cost money while the prototype was in development. No restrictions were made for using frameworks, tools or services that were free to use in development, but cost money once they are used in production.

8.2. Architectural tactics

To fulfill the requirements we had decided on, we used several architectural tactics. This sections show the tactics we used, and they quality attributes they are related to.

8.2.1. Modifiability

High cohesion, loose coupling

To achieve modifiability, we tried to increase the cohesion in our code. By making sure all related code is together, and grouping code that contribute to the same functionality together. We also tried to reduce the coupling, by making modules and component as independent as possible. The components we made would ideally function without requiring any knowledge of other components in the application. This gives us the possibility of easily replacing a component with a new one. As long as the service the component delivers is the same, the way it is done is irrelevant for the rest of the system.

8.2.2. Usability

Consistency

To make it easier for the users we tried to be consistent on colours and shapes. This way users can recognize elements and make interaction with the application more easily.

Follow standards and conventions

Designing our application with respect to standards and conventions the usability will increase. For instance the use of a back button that the users can recognize from other apps helps the user experience.

System feedback

Most actions performed by users should trigger some type of feedback for the users, e.g. when a button is pressed, it should move a little, or momentarily change colour, to indicate it was pressed.

8.3. Architectural Pattern

After evaluating the architectural drivers we decided to use the patterns described in this section.

8.3.1. Model View Controller

Model view controller is a well documented an widely used system architecture. It consists of three parts: a model, a view and a controller. The model contains the data, which in our case would be all the information on the app transferred from the database. The controller contains all the logic of the application and interacts with the model. Finally the view is the visual component

This data, or parts of it, is show to the user in the view. The view contains what the users can see on their device, and is a representation of what can be found in the model. For our application this could be the scrapbook view, and the distribution of students in the GetToKnow feature.

The user, uses the controller, which is code that can be executed, by using buttons or other functionality on the app. An example would be when a students adds a new post to the scrapbook, or when a teacher changes the pairings in the GetToKnow feature. These updates are then sent back to the model, which updates the view to reflect the changes made by the controller.

8.3.2. Client-server

Our application also implements Client-Server architecture.

The apps running on mobile devices are the clients and the server is Firebase. Firebase has predefined protocols for accessing information. When a mobile device sends a

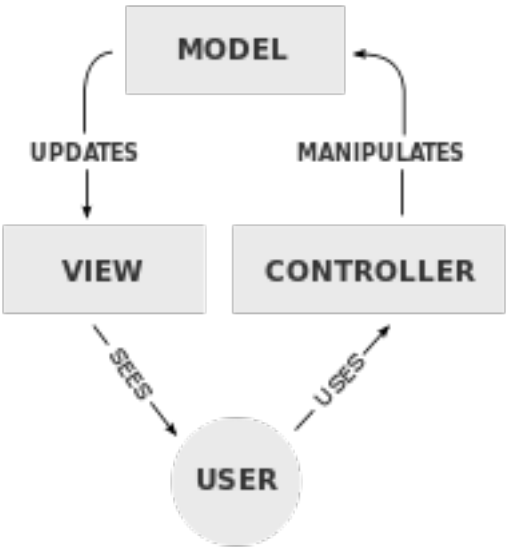


Figure 8.1.: Diagram explaining the MVC pattern [46]
.

request to Firebase, the server will check that it has the proper credentials to access that information. If the mobile device has the proper credentials, the will give it the data over a secure connection. Firebase requires that all communication

We decided that a combination of these architectures would be ideal for our application. It was also advantageous that most group members were already familiar with both MVC and client-server.

8.4. Architectural Views

To be able to view the system from different perspectives we created views based on the 4+1 view model [22] (see Figure 8.2).

8.4.1. Architectural drift and architectural erosion

When designing a system architecture, it is very common to have changes during the project. Architectural drift occurs when there is changes in the planned architecture. Due to unforeseen events it may be ideal to change the planned architecture to some extent. The new planned architecture has drifted away from the original plan.

Architectural erosion occurs when the architecture that is implemented differs from the architecture that was planned. The developers may need to use certain hacks, or poorly implement the architecture, which causes the system architecture to be different than the intended architecture.

Because of these factors we made our diagrams high level, and avoided unnecessary

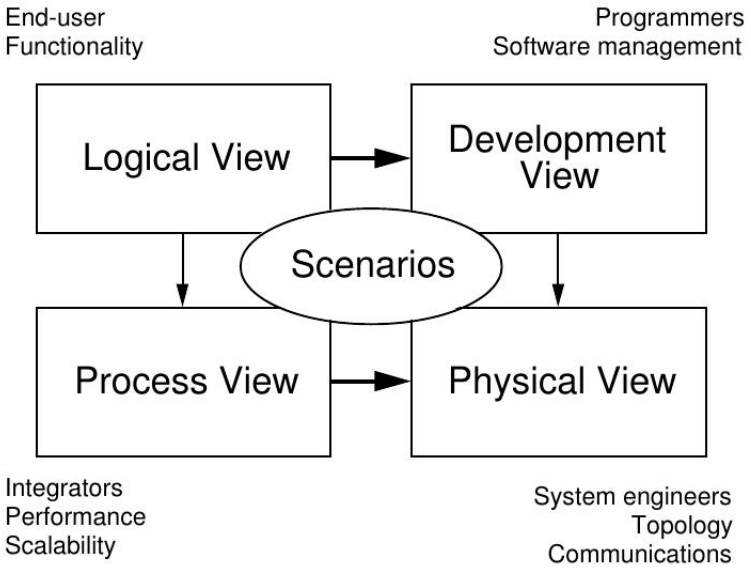


Figure 8.2.: Diagram of the 4+1 view model [22]

details as changes might have happened during the implementation phases of the project. The diagrams we have made were create at a level of abstraction where we expect no architectural drift to take place.

8.4.2. Logical View

The purpose of the logical view is to display the functionality the system provides to the users, as well as help the developers implement the system. We chose to create a class diagram, that shows the structure of the system, and relations between components. We focused on including the most important important classes and methods, as this diagram was supposed to show which classes are related and what the classes do. Details and certain methods or fields may be changed during the project. Additional helper classes might also be added, but the class diagram (Figure 8.3) should give an general overview of the system, in addition to function as a blueprint for what the developers will implement.

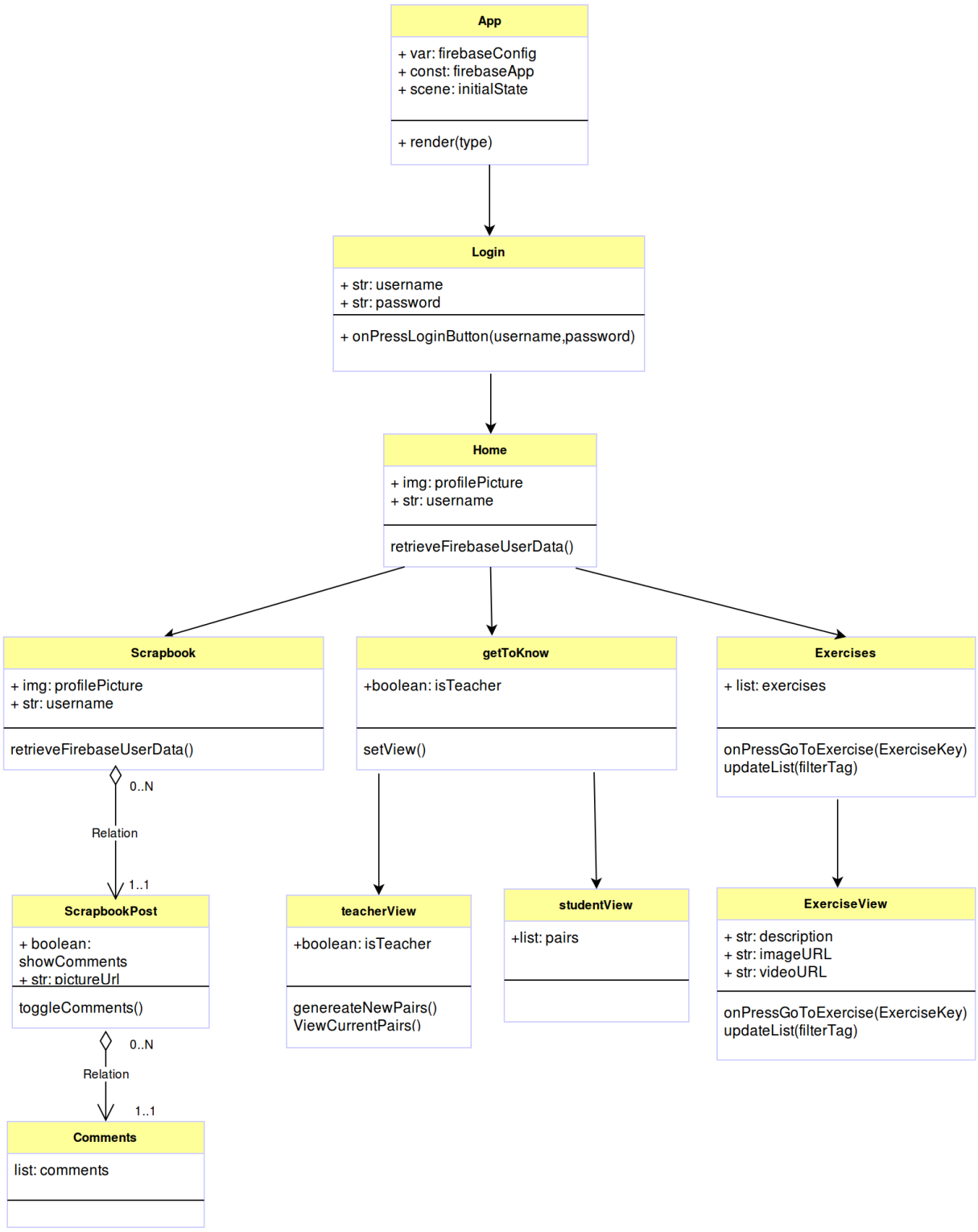


Figure 8.3.: Diagram showing the most important classes, fields and methods

8.4.3. Development View

The development view illustrates the system from the developers viewpoint. Our development view includes a diagram displaying our main components, and a diagram explaining the role of react native.

As mentioned earlier, we chose to use the react native framework. This allowed us to write code for one application in JavaScript, instead of using swift/objective-c for iOS, and java for Android. Below we have included a figure published by SmashingMagazine.com explains the functionality of react native[40].

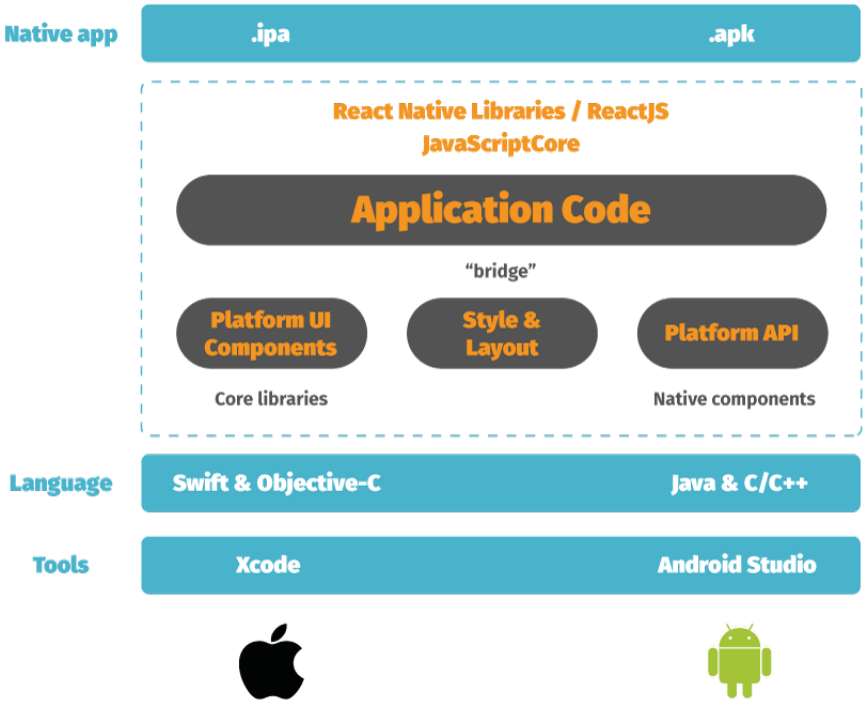


Figure 8.4.: Diagram explaining React Native functionality

The next diagram shows the main components of the system, and the structure of the system concerning use of React native and Firebase.

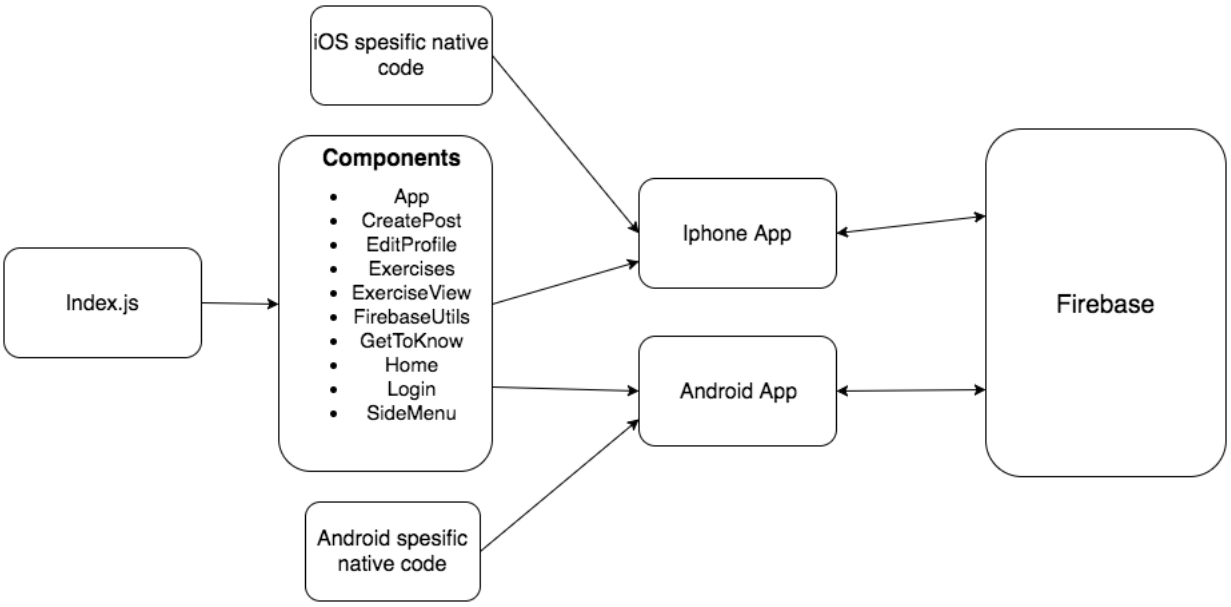


Figure 8.5.: Diagram showing the main components of the system

8.4.4. Process View

The purpose of the process view is to show the dynamic aspects of the application, and illustrate how the system works at runtime. This is done by illustrating user processes, presented as activity diagrams.

Overview

To give an overview of the system we chose to include a diagram that shows navigation through the application, and the diagrams for the most essential activities in the application.

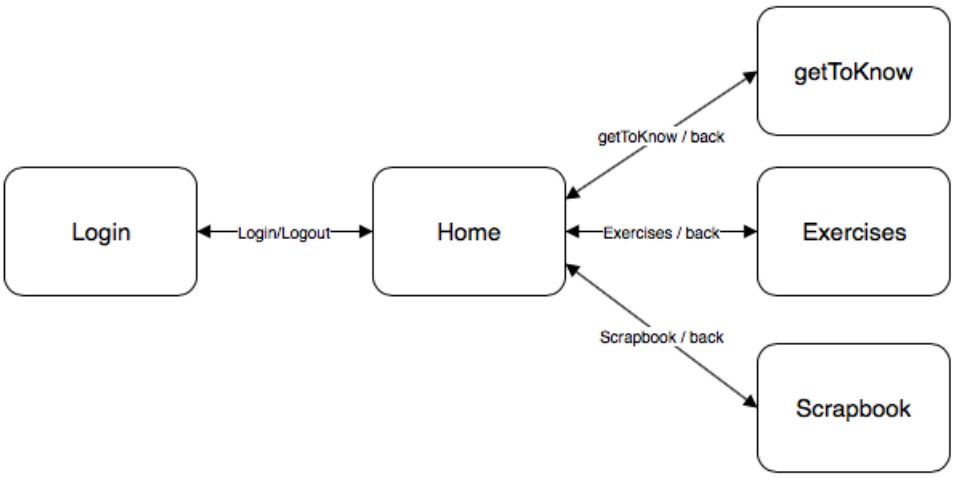


Figure 8.6.: Diagram showing the navigation of the application

Notation and modeling

For the specific activities the notation we used was UML. In the activity diagrams black circle represent the initial state, rounded rectangles represent actions, and diamonds represent decisions, and encircled black circle represents the final state. The decisions are purely yes/no, which means that only two results are possible. The red bars indicate a fork/join. This implies that activities are optional. They can be done in parallel which means the order they are completed in is irrelevant.

The activities we chose to model, were selected because they represent how the most important functionality of the application is used. The activities we chose to model were:

- A teacher creating GetToKnow-pairs and selecting an exericse.
- A student viewing the current GetToKnow-pairs.
- A student posting on the scrapbook.

The first diagrams shows how users interact with the system regarding the GetToKnow functionality, where the teacher creates new pairs that are displayed to the students. The teacher can also select exercises that the students will do.

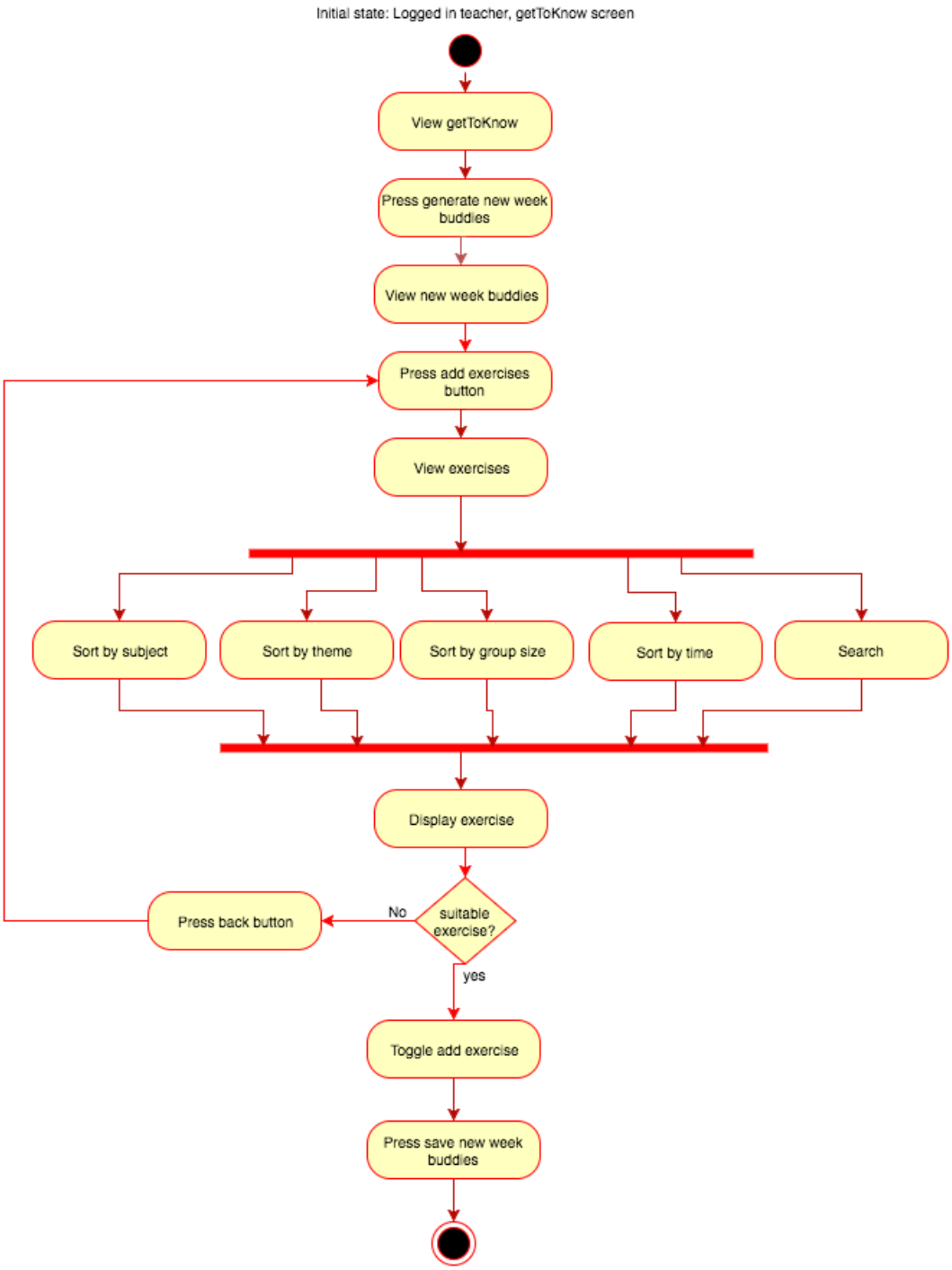


Figure 8.7.: The diagram shows the process of creating new week buddies and selecting exercises

The next diagram shows the activity of a student viewing their new week buddy they

have been assigned to by the teacher.

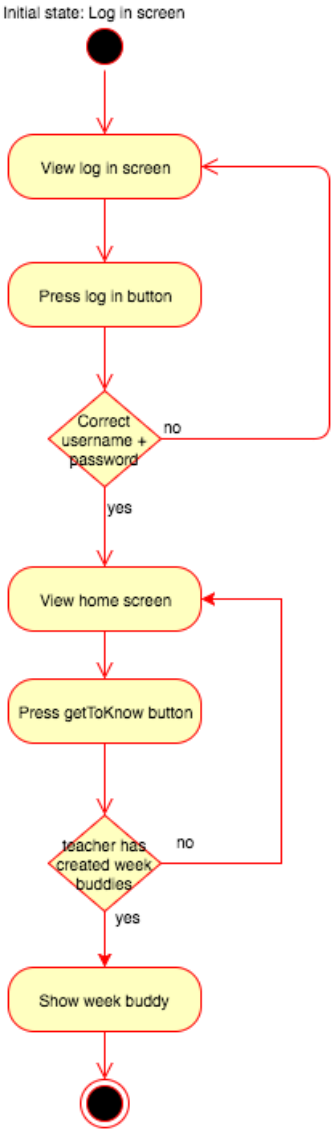


Figure 8.8.: Diagram of viewing your week buddy

The final activity diagram shows the workflow of a student creating a post, and posting on the scrapbook.

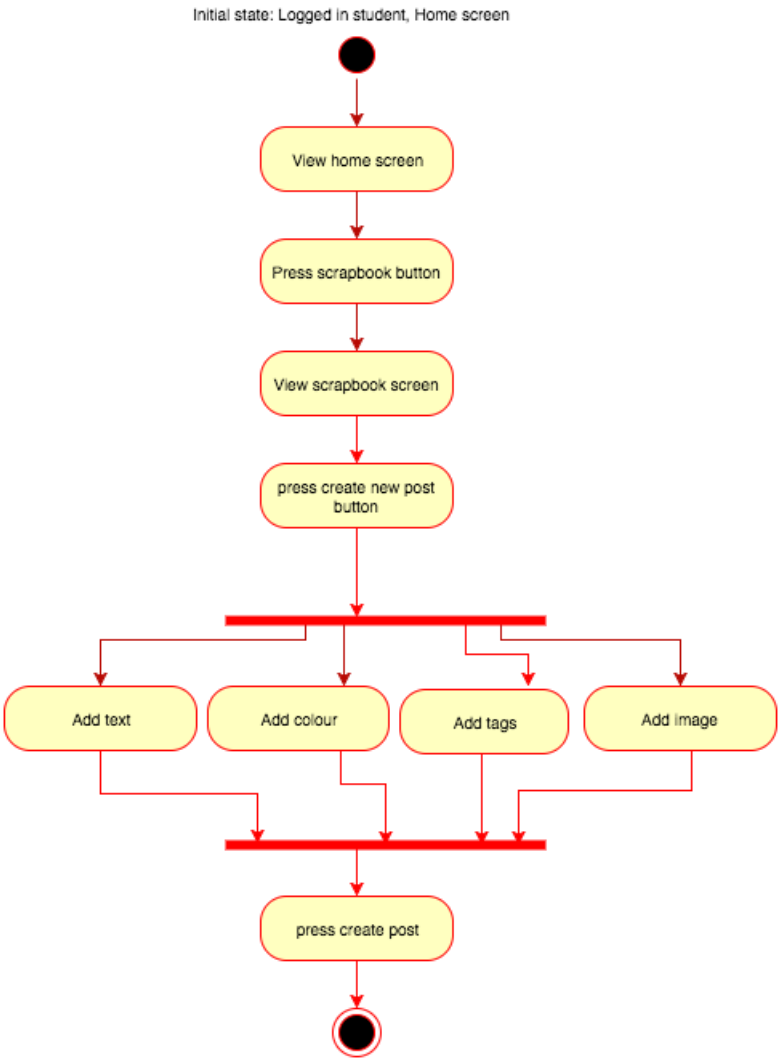


Figure 8.9.: Diagram of posting on scrapbook

8.4.5. Physical View

The physical view shows the physical deployment of servers, services, devices and interactions between these in the system. For our system the most important part of the physical view is the one related to Firebase. Since Firebase is a Google service we have little knowledge of the physical architecture except for the fact that it runs on Google servers, and that it makes use of Google cloud storage for storing media. [14] Since the customer planned to make use of Feide and a web-applicaton at a later date we also included that in our physical view.

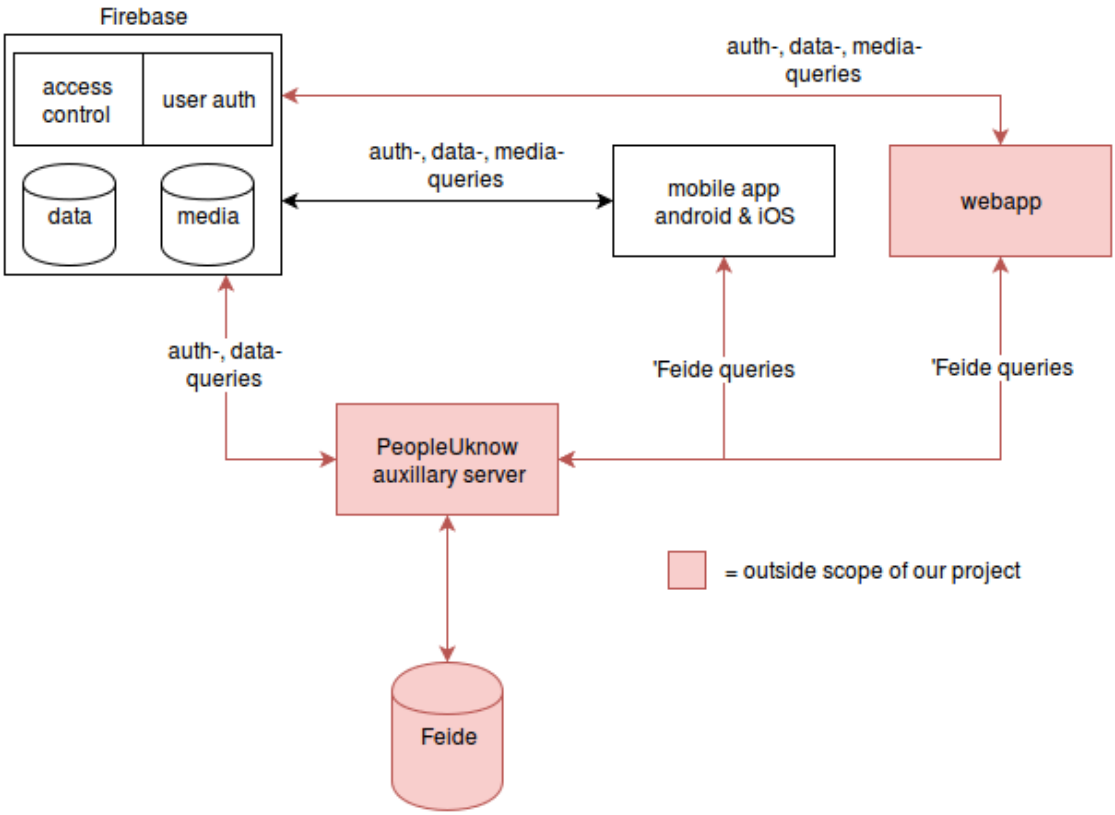


Figure 8.10.: High-level physical architecture of our system. Parts of the physical architecture outside the scope of our prototype is marked in red.

8.5. Database structure

As described in Section 7.4.1 Firebase makes use of a NoSQL database. Unlike a normal relational database which can be modeled as a graph, the NoSQL database of Firebase must be modelled like a hierarchy. We organized our data in such a way that all class related data was grouped first by type at the highest level of the hierarchy, and then by either class or user depending on the type of data. Figure 8.11 shows the hierarchy we used for our prototype. Accessing data in the database is then done in the same way as objects are accessed in a file-system; Each level in the hierarchy is separated by a forward dash. Accessing user-data in our database was then a case of accessing the object at path `/User/<user id>`.

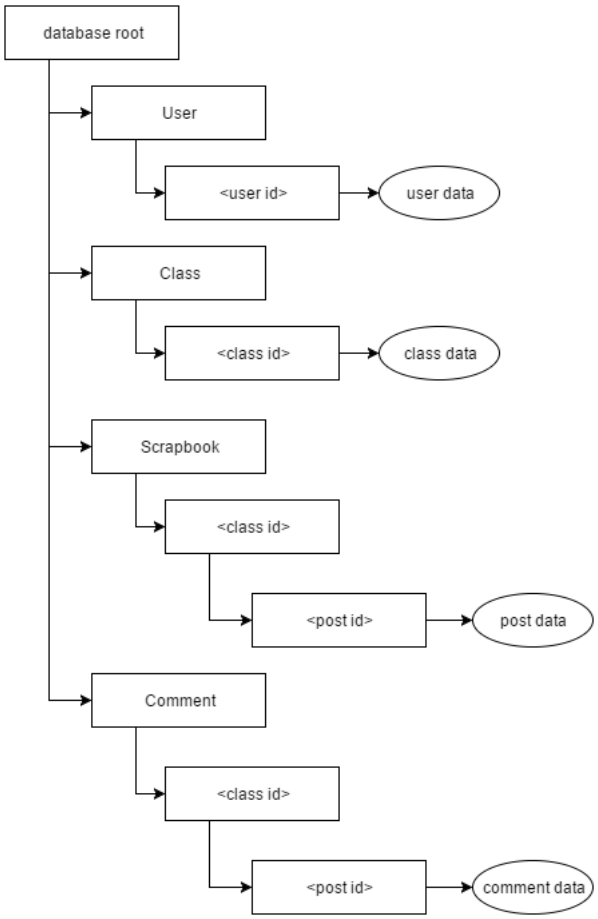


Figure 8.11.: Modelling of our database structure as a hierarchy: Names wrapped in brackets represent variables in the hierarchy.

9. Software Security

In this chapter we describe the security aspects of the system.

Security was a big concern during the project as the users of the app could include children. The app would also store sensitive information about users such as pictures, names and activity data from app usage. To ensure that our app was built with proper security it was important that we considered all known attack vectors, threat agents and abuse cases that are relevant to the app and its back-end.

For analyzing the security requirements we used "7 Touchpoints of Secure Software" by Gary McGraw [27]; specifically we did the touchpoints: Doing risk analysis at the design and architecture level, creating abuse cases, and creating security requirements. We also played Protection Poker [45] in order to prioritize which functional requirements were most important in regards to security.

9.1. Threat Modelling

Threat Modelling is important to understand what are the assets we are protection, who are the people we are and how to protect us against the attacker.

9.1.1. Threat Agents

First we tried to identify the different agents that wanted to attack our system. Those agents could have different experience, motivation, and maybe no intention. The threat agents are described in Table 9.1.

Attackers	**Description**
Accidental Discovery	Regular users accidentally discovers a security flaw. It is not their intention to harm the system.
Automated malware	Automated system that is designed to gather sensitive information or exploit the system.
Motivated attacker: Employees, competitors, enemies, malicious users	These attackers are motivated by an ulterior motive that leads them to put in large amounts of effort in order to exploit the system. These threat agents are often experienced and know of common ways to exploit a software system

Table 9.1.: Threat Agents

9.1.2. Architectural Risk Analysis

For the architectural risk analysis we mostly focused on the architectural choices made in Chapter 8. Most of our concerns were on how Firebase secures the data it stores, this is because the application itself does not provide any real security, and it is possible for attackers to modify it to bypass any security measures. In order to analyse the security in Firebase we used the categories identified in the STRIDE threat model [28] and tried to identify what threats were relevant for each category. Table 9.2 shown the risks we identified associated with the architecture

Risk	**Category**
An attacker extracts data directly from Firebase	Information disclosure
An attacker extracts stored media directly from Firebase	Information disclosure
An attacker performs a brute-force attack directly on Firebase Authentication API	Spoofing identity
An attacker gains access to the administrator panel on Firebase and modifies data	Tampering with data
An attacker performs unintended database requests to modify data	Tampering with data
An attacker performs a denial-of-service attack on Firebase	Denial of service
An attacker performs large amount of media requests to increase bandwidth cost for customer	Denial of service
An attacker intercepts traffic between the application and server to extract sensitive data	Information disclosure

Table 9.2.: Architecture security risks: Each risk is associated with the relevant category in STRIDE

9.1.3. Abuse Cases

To see what an attacker can do to our system, we created an abuse case diagram, which is shown in Figure 9.1. The diagram shows the application, student, teacher and attacker. The attacker can be some of the threat agents from Table 9.1. The student and the teacher are using the application, while the attacker are threaten the system by exploiting the vulnerabilities of the system. The application can also be a threat for the user, since it can store data insecurely and have insecure authorization such that the attacker might steal login credentials [32].

Because of limited of spaces we could not include everything in the abuse case. For instance we are aware of that users can also be a threat for the system.

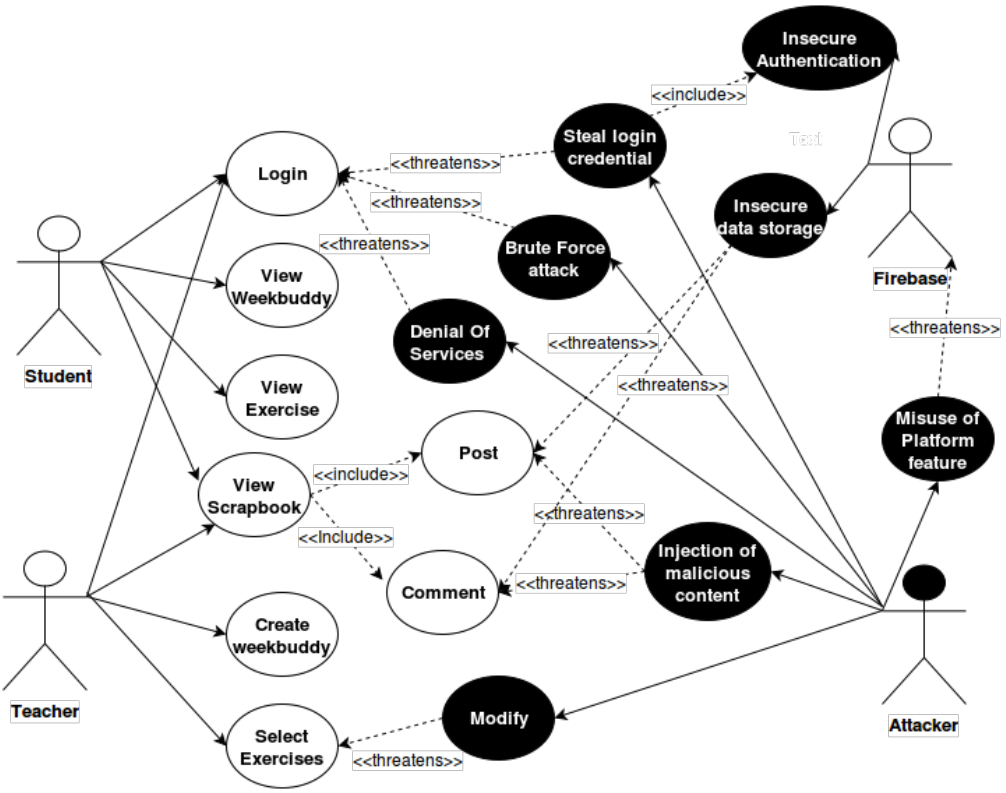


Figure 9.1.: Abuse Case

9.2. Protection Poker

Protection Poker is a gamified method to perform security assessments on software systems [45]. This game is almost as the same as planning poker. The main difference is that planning poker is used for time estimation, while protection poker is used for the assessment on the software.

When we were about to start this game, we got help from two security experts from SiNTEF, Anne Tøndal and Martin Gilje. They were helping and facilitating us to understand how to use this in our project development.

9.2.1. Assess Security Risk

Business Assets

The first thing we needed to do was to identify the important assets for this project. This helped us to understand what a hacker might want to steal. We had a group discussion on what was the important assets by going through all the information assets, software assets, physical assets and services the system had. The result from this are listed under:

- User information (username, hashed passwords, metadata)

- User-uploaded media (pictures, video, texts)
- Sourcecode
- Database
- User credentials
- Firebase
- Youtube
- Exercises
- Class data

Value

It is not only enough to identify the business assets, but we also needed to give them a value. It was important to know which one of them we should be prioritizing since we did not have infinity amount of time and resources. The first thing we needed to do then was to identify which assets was the most important and the least important so we could compare them with other assets.

From the discussion in the group about which one of the assets were most important, we manage to identified that Firebase was the most important assets(Table 9.3) and Exercises was the least important asset. Firebase got the value of 100, which is the highest value. Exercises got the value of 10, which is the lowest but it is still important asset, but lesser important when comparing to the other assets(more are described under).

It was now time to give value to the other assets too. This were performed by using the protection poker cards we got from SiNTEF. The cards had values: <10, 20, 30, 40, 50, 60, 70, 80 and 100. All in the group needed first to choose one of the cards, then everyone needed to show the cards at the same time. If the cards showed different numbers, then the group had to discuss why they chose that value. It continues until the group had gotten a common assessment of the asset.

Asset	Value
Firebase	100
User-uploaded media	80
Database	80
User credentials	70
User information	60
Sourcecode	30
Youtube	30
Exercises	<10

Table 9.3.: Protection Poker Score Sheet- Assets

- **Firebase:** consists of the authentication, pictures, users, and database. This is the important and critical assets. If an attacker can get access to Firebase, she/he can basically delete everything or just steal, and modify the application.
- **User-uploaded media:** is all about what the users are uploading which are posts, pictures, movie etc. This is data, have also high value since it involves pictures and video which can be used for identifying users. But this has not higher value than Firebase because it is a subset of Firebase. If the attacker manages to get access to Firebase, the hacker has then taken over the user-upload media too.
- **Database:** contains data relating to the users, and the class structure. If an attacker takes over the database he/she can delete every user, retrieve all the information in the database, or insert something malicious etc. This is also a subset of Firebase. When an attacker gets access to the database, the attacker has not got everything in Firebase, only the data in the database.
- **User Credentials:** are emails and password, which gives access to users. It is important to ensure that hackers do not get access to users' credentials. When we compare user credentials with Firebase, Firebase is more critical to protect than user credentials. Imagine an attacker gets user credentials. The only thing the user can do is all the regularly user can do (such as student privilege and teacher privilege). But if the hacker gets into Firebase, the hacker gets all the user credentials, which can do bigger harms.
- **User Information:** is user's name, statistic over the activities, week-buddy etc. but we had decided to not include information on birthday because this has no function for users in this application. We discuss also that user information will not disclosure anything that can be helpful for the attacker to use in order to figure out password etc. If an attacker really wants a user's information, the hacker could just stalk the victim on Facebook or at school to get more. But still, we need to protect the user information because of the law in Norway, "Lov om Behandling av Personopplysninger" from §13. Information Security [25].

- **Sourcecode:** is the fundamental component of the program. Leaking the source code is not a problem since it can be easily found. One thing the attacker can do is to use this code to impersonate the application, therefore it is still an asset we need to consider.
- **Youtube:** asset is the PeopleUknow's account on Youtube. All their exercises will be posted from that account and we need therefore to consider this. It will be difficult for an attacker to get this since we assume that the security of Youtube is good. Youtube is used by billions of people [50], they should therefore be able to protect their users by having a good security system.
- **Exercise:** is video exercises posted by PeoplUKnow Youtube account on Youtube. We see this asset as not important since the assets are already out there on the internet, and if someone wants those exercises it is really easy for them to find it on Youtube.

Exposure Value

Exposure value describes in which degree the requirements that are going to be implemented, will affect on how exposures these assets will be. We discussed in the group how the requirements could influence the exposure. Also, we needed to take into account what aspect of the system asset could be breached and if the attackers could get full access, read only or it could affect the availability of the assets. To find the exposure values we did the same as we did for the asset values; each chose a protection poker card, the card got presented for the group, the value got discussed and re-voted if needed.

Figure 9.2 shows the calibration of asset and exposure. It was important to start with highest and lowest, to be able to prioritise between different requirements and to be able to spread the numbers assigned. 100 is given to the requirement that have highest exposure for this project, and similarly <10 was given to the requirement that had low exposure for this project. According to [45], this was to avoid to rate every requirement as high risk, which would make it difficult to prioritize within the project.

Requirement F2, had the lowest exposure value and F1 got the highest exposure value. More detailed on how we found the exposure value for each in Table 9.4 can be found in the Appendix C.

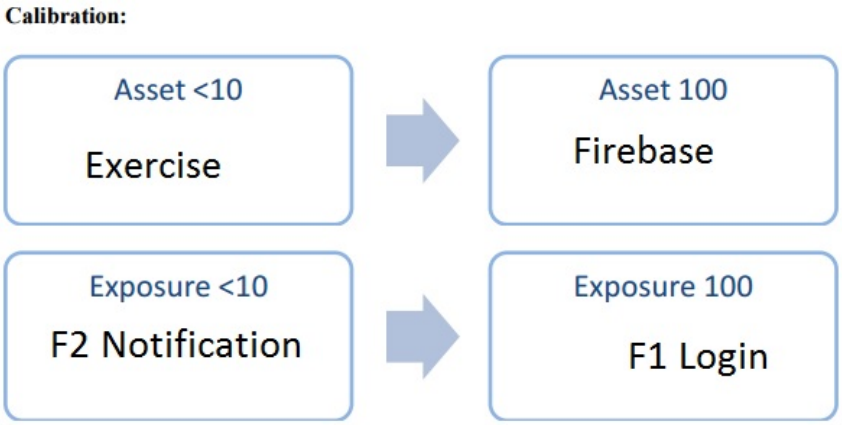


Figure 9.2.: Calibration of Asset and Exposure

Calculate Risk

Risk is calculated by summing up the value of assets that could be exploited multiply with exposure value.

$$Risk = (AssetValue) * (ExposureValue) \tag{9.1}$$

The Table 9.4 shows the risk which is the overall score[1]. exposure value is from playing protection poker, asset value is summed up from all asset that are involved in requirement(which can be found in the Appendix C and the overall score is the risk.

Table 9.4.: Protection Score Sheet

Requirement	Exposure Value	Asset Value	Overall Score
F1	100	340	34000
F2	<10	280	2800
F3	50	360	18000
F4	60	190	1140
F5	80	290	23200
F6	30	350	10500
F7	60	350	21000
F8	50	250	12500
F9	30	250	7500
F10	30	350	10500
F11	50	350	17500
F12	40	350	14000
F13	<10	110	1100

[1]For more details see the table in the Appendix C

Compare Risk

From the Table 9.4 we could identify which requirements have higher priority in terms of security than others. In order to better group the requirements based on risk, we distributed the requirements evenly into three groups as shown in Table 9.5, extra care should be taken when implementing the requirements with high or medium priority.

Priority	**Requirements**
High	F1, F5, F7, F3
Medium	F11, F12, F8, F10
Low	F6, F9, F2, F4, F13

Table 9.5.: Requirements grouped by security risk

9.2.2. Security requirements

Using the threats identified in Section 9.1 we created security requirements. The security requirements we identified are listed in Table 9.6

S1	All data must be verified before being sent to the database, users should only be allowed to view data they have been given access to.
S2	All images or documents stored by the prototype must only be accessible to the appropriate persons
S3	User passwords should be hashed using up-to-date hashing algorithms and not stored in raw format
S4	Communication between application and external services should be encrypted
S5	It must not be possible to insert data into the application that disrupts the experience for other users

Table 9.6.: Security requirements

Steps taken to mitigate security risk

Since we use Firebase for database, storage and authentication we could rely on their platform for mitigating many of the security risks. According to Firebase they use best-practices for authentication tokens, [15] when we also consider that Firebase is a Google product we assume that the authentication on Firebase is secure.

For encryption between application and Firebase we assume that the connection is secure: Firebase uses Secure Socket Layer with strong keys which make it unfeasible for attackers to eavesdrop on connections. [15]

The only remaining security risk we felt was important to mitigate was preventing users from reading or writing database objects they do not have access to. In order to

ensure correct access rights in the database, we used Firebase's powerful security rules feature: Each level of hierarchy in the database have rules that define which users are allowed read or write to them. [15] Our implementation of these rules are documented in Section 14.4.1.

10. Testing

In this chapter we describe the various testing methodologies used and considered in the development of the application.

10.1. Overview of Testing

We made a plan to cover most aspects of the development process. Our focus was on verifying that we made the right product, i.e. the product that the customer wanted. This was because the prototype we were to deliver was not meant for selling directly to customers, but used for demonstration and refining the design of a later solution. We therefore put more effort into acceptance and user testing than unit testing and performance testing. Table 10.1 describes the main aspects of our test plan.

Test Carried Out	**Stakeholders involved**	**Time of testing**	**Description**
Functional testing	Developers	Every sprint	Test the functionality of each new component. Tests should be conducted as soon as possible, be testing how the component works in the simulator, preferably during the same sprint the components were developed.
Code review	Developers	Every sprint	Evaluate the code we have just implemented, and look for errors or problems.
Usability Testing	Developers and end users	Week 39 and 42	First test will be conducted using paper prototypes and other simple user tests, to test navigation between the features of the app. Test should be conducted as early as possible to help create a design with high usability. Second test will be conducted on a group of students, and will be conducted later, to get input on a more complete version of the app.
Acceptance Testing	Developers and product owner	At the end of every sprint	The acceptance test is conducted with the product owner present, to evaluate the final application.

Table 10.1.: Overall Test Plan

10.2. Unit Testing

Unit testing is method of testing where small parts of the code are tested to evaluate if they work they way they should and are fit for use.

Use of unit testing in our project

Many software projects use automated unit tests to test small parts of the system, and evaluate the coherence between input and output. We decided against unit tests for our project for a number of reasons. First, none of our group members had experience with unit testing using Javascript and React. Second, unit testing becomes impractical due to the unique way our project is structured: Since we use Firebase as a backend, we would have to write all of our unit-tests on frontend code. Almost all of our functions on the frontend run asynchronously, both for network calls and setting the internal state of the app.

We therefore reasoned that unit testing would be extremely time-consuming and therefore not worthwhile. Instead we opted for running functional testing to make sure our solution worked on most use-cases.

10.3. Functional Testing

Functional testing is a process of testing the functionality of the different system components. These tests ensures that the features works as intended from the user's perspective and satisfy the functional requirements.

The functional testing in our project was done by performing manual tests of every functionality of the application in a simulator. These tests made it possible to identify errors and faults in the code by comparing expected test results with actual test results.

Functional Test 1	
ID	FT1
Description	Tries to log in
Date	27.09.16.
Tester	Ivar
Preconditions	App must be running, password and email correspond, and exist in the database
Tasks	1. Type in email 2. Type in password 3. Press the log in button 4. Confirm that you are redirected to home-screen
Result	When the log in button is pressed the user is redirected to the home screen.

Table 10.2.: A functional test from sprint 1. The tests are listed in Appendix D.1

10.4. Code Review

Code review is the process of detecting mistakes through systematic examination of the code with a fellow programmer.

We conducted our code reviews mostly by reviewing pull requests. In our version control system, GitHub, we created a branch for each functional requirement. When a function is finished and ready to be merged into the main branch, a pull request is created. A fellow programmer in the group is then given the job to review and approve the new code before being merged.

10.5. Usability Testing

Usability testing is a technique where you test your product on end users to evaluate the usability of your product. This is a valuable technique because it gives you direct feedback from the intended end users that can be used to evaluate and improve the product.

In our project we focused on formative, and summative usability testing.

10.5.1. Prototyping

The quote "just enough prototyping", by Gillian Crampton Smith, was kept in mind while we made our prototypes. We made most our prototypes on paper, by drawing our intended layout on a print of an Iphone 5 sketch. In addition we used an online tool called Marvel [26]. This is a web application that allows for rapid prototyping of user interfaces on mobile devices. The different screens are designed using images and pressing on different parts of the screen can be configured to navigate between images. This

makes it possible to take advantage of implicit knowledge in the users, and recognition from previous experiences with mobile application, since buttons and colours can be more realistic and similar to other apps, compared to hand drawn images. However, prototyping with Marvel, was more time-consuming, and for most tasks paper-prototypes was sufficient.

10.5.2. Formative

Formative usability testing should be done in the early design face. It is sufficient to use paper prototypes, or something similar. This is done to get insight from end users, which can be used when improving the design.

Horizontal Prototype

The goal of the first test was to evaluate the navigation and layout of the different features in the application. Since the goal was not to test in depth functionality of one feature, it was sufficient and time-effective to use paper prototype. The Figure 10.5.2 shows one of the prototype we used for testing (more can be found in Appendix D.2).

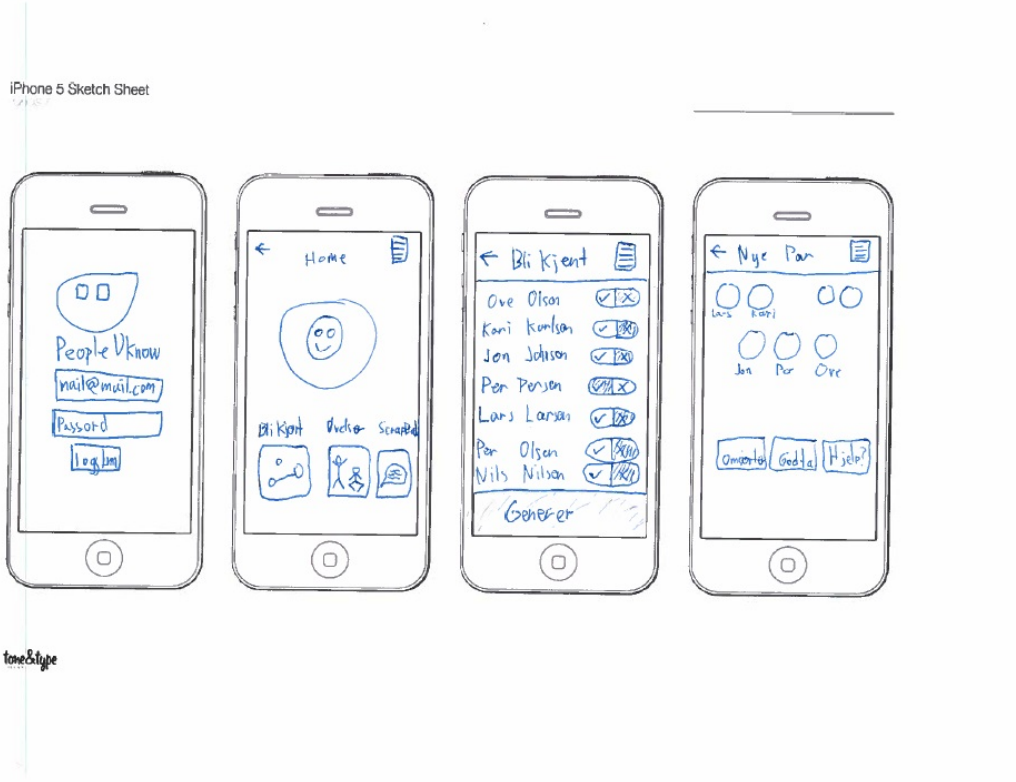


Figure 10.1.: Paper Prototype: Application

Vertical Prototype

As we began implementing the different functionalities, e.g. GetToKnow we created more detailed tests. The goal of these tests was to evaluate the usability of the different functionality, and determine if users would struggle to use the application the way it was intended. For these tests the test subjects were only given access to some of the functionality in the app, but a prototype with more details and in depth functionality.

Participants

For formative testing we recruited three students from secondary upper high school in order to test the system. They are the end users of the system PeopleUknow want us to develop. Their age are between 16-19 years old.

We also manage to recruit 3 adults and 6 students to test the system.

Method of Formative Testing

In the first weeks we did formative testing. The goal of the test was to get feedback that could help us decide how to improve the design and increase the usability. For this testing we tested two groups of participants. Since it is the students and teachers that are the important one for the system, it was created two usability tests. Before the testing, it was important to create test plans so we focused on the important things of the system and that we could get as much feedback from the participants. The test plans that were created can be found in the Appendix D.3. The test plans are based on our user stories.

When all test plans were in place, it was time to recruit participants. It was also important to inform the participants about the purpose of the test, to make sure they understand that we're testing the system and not the users. The participants got also explained what the system was and he/she should not be worried about doing anything wrong. The participants were ask perform some tasks and explain why they were choosing to do things in that way. A test manager was observing the participants and took notes.

When the interaction was complete, the participants was ask to evaluate the system. This was performed by asking the participants about their impressions of the system, what was difficult and what could been improved.

Scenarios

The participants were asked to do some tasks. These tasks were simple and based on our user stories. The tasks we asked them to do is listed below:

- Log in with email and password
- View the exercise
- Overview of the application
- View the partner

- View the scrapbook
- Comment on the posts
- Sharing on the scrapbook
- Search on specific theme

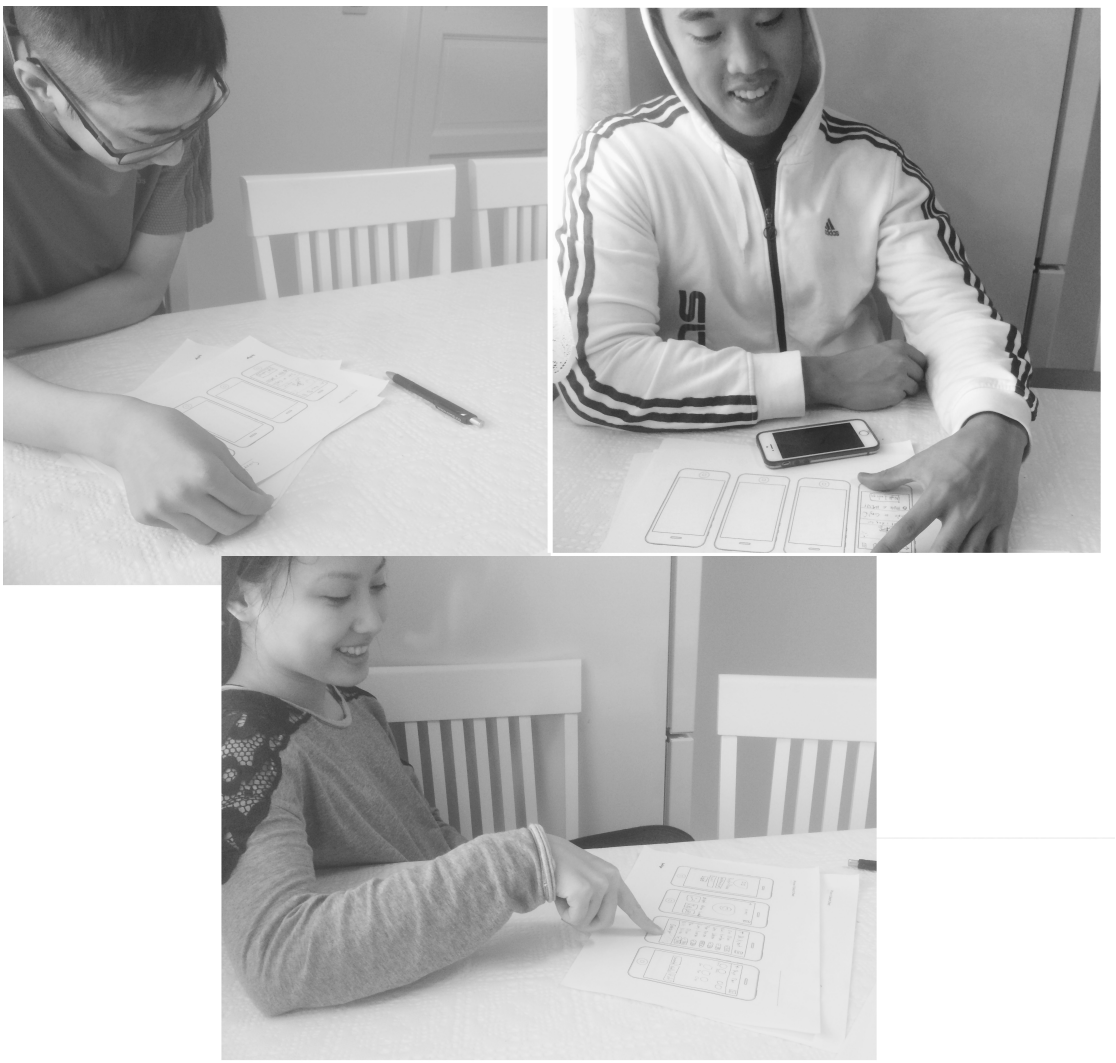


Figure 10.2.: User Testing

Feedback from Students

The list below shows the feedback from the students. More detailed feedback can be found in the Appendix D.3.3.

- The login page was easy to understand. It looked like a regular login page.
- Could maybe include a progress-bar or something in the exercise view?
- Scrapbook Icon looks like a chat box. This looks like where you can chat.
- Make the comment box more clear.
- The menu bar which is right up in the corner seems like a place where the search function is.

10.5.3. Summative

Summative usability testing is done in the later stages of the development phase, using a more complete version of the product. The results from this test can be percentage of tasks that were completed, and time per task. If the design is changed to improve the usability, a new test can be conducted, and the new results can be compared to the results from summative usability test.

System Usability Scale

In order to measure the usability of a system, System Usability Scale (SUS) can be used. SUS measure how well a system works and how good it is received by the users. It consists of ten-item scale which gives a global view of subjective assessments of usability and the users have five option to answer. After a user have tested a system, the user will get the SUS schema to evaluate the system. In Appendix D.5 shows the SUS schema that was used for this project.

Participants

In summative testing we managed to recruit 6 students and 3 adults by help from PeopleUknow. The age of the students was in between 15-16. We used the adults as teachers for the testing.

Method of Summative Testing

For this test we tested on a group of students. This was to evaluate the near-complete design under realistic condition. The test was performed by first informing everyone that we were testing the system and not them. After that we had a person that observed the participants (as in the formative testing).

In order to see if we met the requirement of what the system are supposed to do, we tested the system on a group of student. After testing we needed to collect information from the students, which we did with the System Usability Scale (SUS).

Scenarios

As same for summative, the participants were asked to do some tasks we had prepared for them. Below is a description of the different tasks they were to complete:

- Login with email and (wrong and correct) password
- View the post
- Comment on a post
- Create a new post with styling
- Find your week buddy
- View your exercises with your week buddy
- Take a new profile picture
- Log out

Feedback

From this test we got a lot of positive feedback. Most of them said it was a good application, and some of them said they would have used it. Feedback on what we could improve with the application is summarize under[1].

- Do something with they keyboard, since the keyboard hide the password field it was difficult to type in password.
- Give the menu button a name or description.
- Wants something to happen when clicking the pictures.
- Should be possible to see chosen tags for the post.
- Language conflict between English and Norwegian.
- Menu button should be place where it seems natural becuase it hided the status bar for Iphone.

[1]More feedback can be found in Appendix D

After the users tested the system we gave them a SUS scheme so they could evaluate the system.

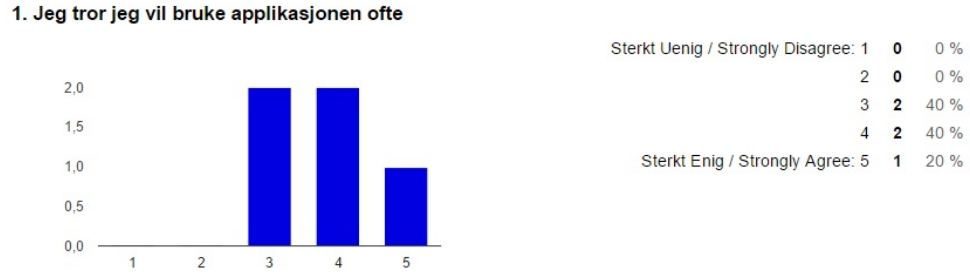


Figure 10.3.: I think that I would like to use this system frequently

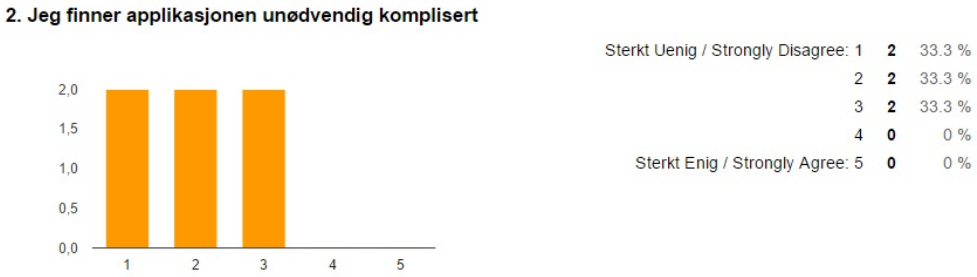


Figure 10.4.: I found the system unnecessarily complex

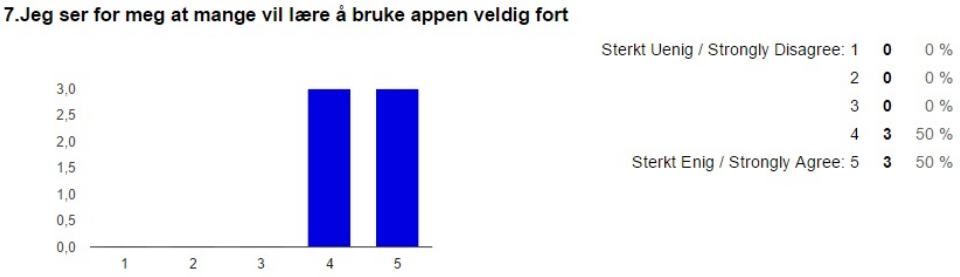


Figure 10.5.: I would imagine that most people would learn to use this system very quickly

More figures can be found in the Appendix D.3.4.

SUS score

The SUS schema can be used to generate a simple numeric score between 0-100. Although this result is simple and one dimensional, it is useful because it is easy to compare to

other systems, or different versions of the same system.

Calculating the score

After the test participants have answered the ten questions, the sum of the answers is combined and divided by number of participants, to get the average answer for each question. For odd number questions, the score is average value minus one. For even numbered questions the score is five minus the average score. The final scores for each question are then summed, and multiplied by 2.5.

Our score

According to research on evaluating sus scores, [36] [4], the average sus score is about 68. Our sus score was 79.15 which according to the research correlates to the adjectives good and excellent.

10.6. Acceptance Testing

Acceptance testing was done with the customer at the end of each sprint. The acceptance test was done as a part of the sprint review. We reviewed the the features that was implemented and got feedback from the customer. We got useful feedback in terms of design and some things that could be done differently to make it more intuitive.

11. First Scrum-sprint

11.1. Sprint planning

At the start of the sprint we met with the customer to agree on what would be done during the sprint. We first presented the functional requirements we had identified and agreed on what we would try to accomplish during the whole project, after some discussion we agreed to focus on some core features of the app, specifically the student pairing, scrapbook and exercise features. Since it was so early in the project we wanted to focus on learning the development tools and getting the fundamental features implemented correctly. As such we agreed to spend most of our development efforts on setting up the login.

During this sprint we would also prepare a paper-prototype for use as part of the formative user-testing, the details of this are described in Chapter 10

11.2. Sprint goals

Based on the sprint planning described in Section 11.1 we decided on the following goals:

- Setting up the report structure
- Get a working understanding of React Native
- Create prototypes for user-testing
- Get a working understanding of Firebase
- Add the ability to log into the app
- Properly configure the backend connection on the app
- Implement a home screen

11.3. Sprint Backlog

We decided to limit ourselves to a relative small number of requirements because we were very inexperienced with the framework and tools at that time.

We also expected to perform user testing in the next sprint. Therefore, we also gave ourselves time to properly plan this during this sprint. For our backlog on Trello, it looked something like Table 11.1.

Name	Estimated	Actual
Create login page	20	18
Integrate login with Firebase backend	50	40
Create prototypes for user-testing	20	30
Create home screen	40	31
Configure backend connection	20	32

Table 11.1.: Sprint 1 backlog: Estimation and actual effort are in hours.

11.4. Result from the Sprint

11.4.1. Login functionality

The backend implementation with Firebase was done by implementing the official Firebase API provided by the developers. Configuring the backend communication was then done by initializing the Firebase API with the id of our Firebase project. We configured default users in Firebase which we used for testing during development. We used design sketches from the customer to implement the view for the login screen. We also made sure to add error messages when the login failed, this was done by integrating a notification framework into our app.

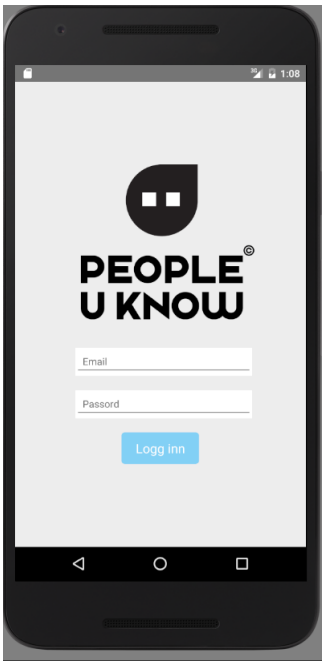


Figure 11.1.: Resulting login screen from sprint 1

11.4.2. Home screen

Once the login functionality was in place, we implemented a basic home screen that would serve as a base for navigating. The home screen includes a profile picture and the username, however since we did not have the ability to upload images to the database, the profile picture on this screen is a placeholder. The screen also includes placeholder tabs for showing how many posts that have been shared on the scrapbook, but since the scrapbook hasn't been made yet they are also placeholders.

We also added a non functional navigation bar to the top of the home screen, according to the design sketches.

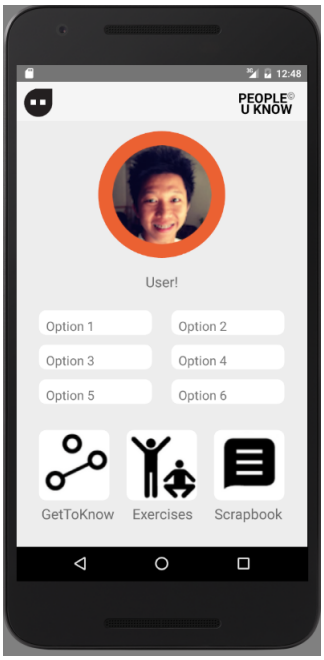


Figure 11.2.: Resulting home screen from sprint 1

11.5. Customer Feedback

At the end of sprint 1, the customer got to see what had been done so far. The customer was happy with the result from this phase. We got feedback from the customer on some styling choices made on the home screen: They wanted more spacing between the top of the home screen and the top of the app, more space between options on some aspect-ratios and that the font-size could be smaller.

Sidebar menu The customer also wanted us to swap the position of the icon and the text in the navigation bar. Pressing the icon would then lead to a sidebar with additional options being displayed. We agreed to implement this during a later phase.

11.6. Sprint Retrospective

What went well?

- We were able to complete all tasks we planned to do. Tasks were completed unexpectedly quick. Firebase was easy to work with, and most coding was done relatively quickly.
- Distribution of workload was good, and everyone in the group was able to contribute.
- Good standard for working in parallel, new branches etc.
- No big merge conflict issues.

What went wrong?

- We should have kept everyone better updated on what each person was doing.
- Some of us had problems with setting up the development environment for Android and React Native.
- Using Facebook to organize the project group was too chaotic.
- We had to reschedule meetings with PeopleUknow and Webstep due to misunderstandings about meeting times.

What can be done to improve?

- Start using Slack. Take all serious discussions there. Keep Facebook for quick questions and notifications.
- Be clear about when and where meetings will take place when talking to the customer.
- Do digital standups each day on Slack: Each person makes a post each morning about what they have done and will do today.

11.7. Sprint Burndown Chart

Figure 11.3 shown the burndown chart for this sprint. Effort remaining each day was calculated by estimating how much work was left on each task and summing the values.

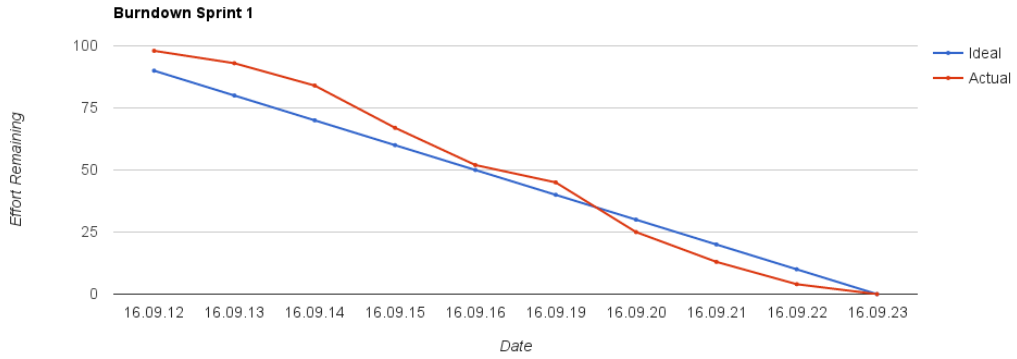


Figure 11.3.: Burndown chart for sprint 1

12. Second Scrum-sprint

12.1. Sprint planning

We met with the customer to decide what we should focus on during the second sprint: Since we got most of the fundamental work done during the first sprint, we decided to focus on getting more actual usable functionality into our prototype; We agreed that we should firstly focus on the scrapbook, since this is one of the tasks the customer was most excited about us implementing. When talking about the scrapbook feature we were informed by the customer that they wanted the ability to upload video to the scrapbook. We were sceptical about our ability to add working video upload to the prototype, so we agreed that we would only make it possible to upload still images to the scrapbook.

We also agreed that we would implement the list of exercises that would be viewable by the teachers, this was because we did not think we had time to implement the student pairing feature this sprint, but would have time to implement a smaller part of it which was the exercise view. We would then integrate the exercise view with the pairing feature in the next sprint.

We were also contacted by people from SINTEF about using protection poker as part of the security investigation of our project. We agreed to perform this task as well during this sprint.

12.2. Sprint goals

Based on the sprint planning described in Section 12.1 we decided on the following goals:

- Play protection poker
- Perform user-testing
- Make a list of exercises that is searchable
- Make a functional class scrapbook
- Have the ability to add images to scrapbook posts

12.3. Sprint Backlog

Our backlog became such as the Table 12.1.

Name	Estimated	Actual
Create component for viewing scrapbook	30	35
Create component for posting on scrapbook	40	54
Learn Firebase access control rules	10	9
Perform user testing	10	14
Create exercise view	30	30
Make exercises searchable	20	45
Create database structure for scrapbook and exercises	15	20
Create component for single scrapbook post	20	27

Table 12.1.: Sprint 2 backlog: Estimation and actual effort are in hours.

12.4. Result form the Sprint

12.4.1. Scrapbook

The scrapbook component was implemented successfully after some initial problems with getting uploading to work with the Firebase API; We spent a lot of our development time trying to fix this issue, and eventually had to contact Firebase directly and report our issue. It turns out that we had encountered an unknown bug in the Firebase API, and were able to create a workaround until the Firebase team could release a patch.
The "show comments" button is a placeholder that would be implemented in the next sprint.

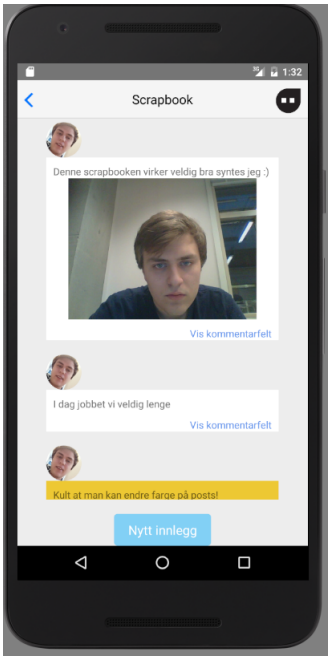


Figure 12.1.: The class scrapbook view that we implemented, showing off pictures, text and custom styling of posts

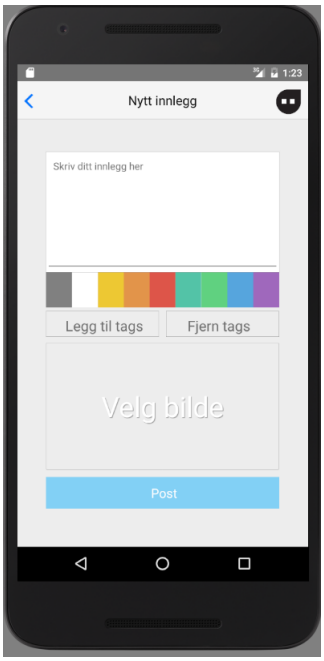


Figure 12.2.: Component for posting on the class scrapbook

12.4.2. Exercises

Exercise list The list of exercises was implemented with a search functionality that updated the list for each letter written/erased, searching both titles and tags associated with each exercise.

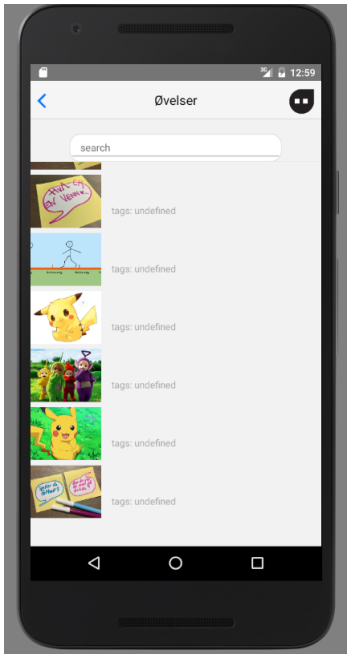


Figure 12.3.: Component for listing of exercises, titles and tags are missing from the view because we changed the database at the time of taking this picture. The pictures are placeholders.

Exercise view The exercise view turned out as planned with working search, video support, and tagging of exercises. We had some issues with getting videos to properly work on Android but other than that the implementation of this feature went smoothly. The product owner had expressed that she had plans for how the view were supposed to look, but the sketches were not yet done, so there was little focus on the design this sprint.

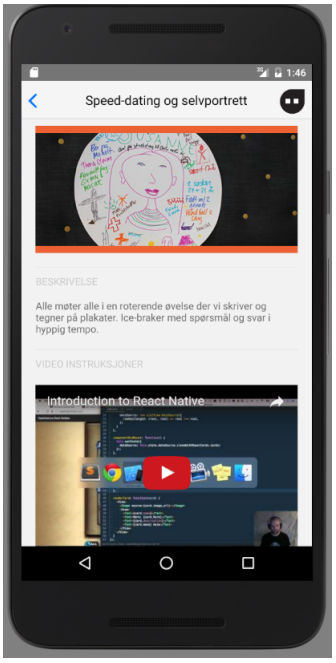


Figure 12.4.: Component for showing exercise description. The video is currently a placeholder, but can be replaced by a youtube video.

12.5. Customer Feedback

During the sprint review we presented what we had worked on so far to the customer. They were happy with the progress so far, but had some additional feedback to some of the features we had implemented:

Exercise filtering
The customer also wanted searching on exercise category and not only with a searchbar, the teacher should be able to filter on topic, exercise length and group size. We agreed that we could modify the exercise to accommodate this in the next sprint.

Scrapbook
Scrapbook looked good according the customer. They also asked us if it would be possible for students to upload videos. We told them that making working video upload and viewing is a very hard task and while possible, would still be too much work to perform during this project.

Exercise view
The customer suggested that the exercises could be group by type so the list could be better structured, this would also allow for minimizing groups so that the list could become shorter. Since this was not a core feature we did not make any promise of

implementing this change, but would look at it if we had extra time.

12.6. Sprint Retrospective

What went well?

- Using slack was a good idea. It was easier to communicate and we got more structure on the communication.
- Merging the different branches
- Meeting with Webstep went well. They gave us 4 hours of their time in each week to help us with things if we needed.They wanted to help and guide us through the development phase.
- Development went generally smoothly, we were good at helping each other out on difficult problems.
- Manage to start on the security aspect by playing protection planning.

What went wrong?

- Used too much time on upload and download of pictures. The reason was that there was a bug in the Firebase API. We tried a lot of things we should have skipped.
- We had some overlap in some task, which we use a lot of time on. Mostly relating to the Firebase API bug.
- There was some sickness in the group, this impacted the development during this phase.

What can be done to improve?

- Can't do anything about people getting sick, we have to respond better to unexpected events such as this.
- To avoid overlap we can create a technical channel on slack, where we discuss technical issues and things, so that we do not do overlapping work.
- Next time we should have the demo ready the day before we meet the customer.

12.7. Sprint Burndown Chart

Figure 12.5 shown the burndown chart for this sprint. Effort remaining each day was calculated by estimating how much work was left on each task and summing the values.

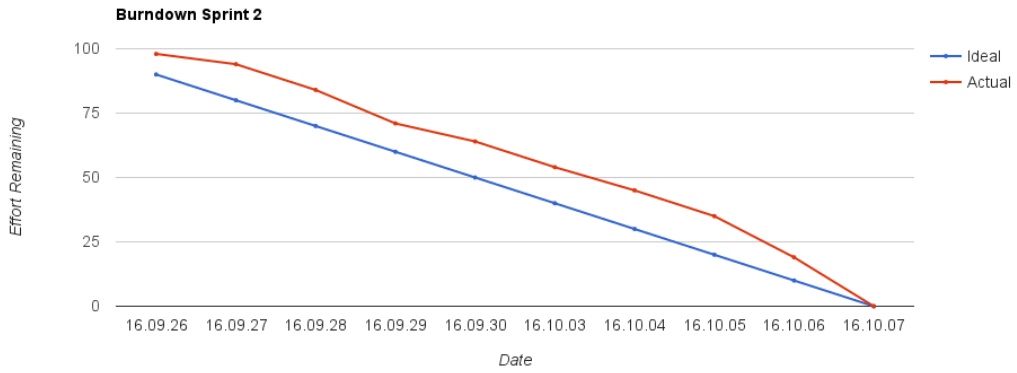


Figure 12.5.: Burndown chart for sprint 2

13. Third Scrum-sprint

13.1. Sprint Planning

During our sprint review with the customer we agreed to improve the filtering of the exercise list. We also agreed to start working on the student pairing which includes viewing and creating such pairs that were part of the GetToKnow feature. We were not sure if we had time to implement the GetToKnow feature fully so we agreed to focus on the the user experience first, and improve the backend in the next sprint if we didn't have time to finish it.

In the previous iteration of the app the profile picture on the home screen was a static placeholder. The customer decided that they wanted the ability to change this picture from inside the app in our prototype, which we agreed that we could implement this during the sprint. The customer wanted the ability to crop the profile images before uploading to the database, we were unsure if we could do that, so we promised to look into it and implement if it didn't take too long.

Finally the customer wanted a side menu that included notification and additional options. We concluded that this was a good place to put the change profile picture option, so we included this feature in our sprint backlog as well.

13.2. Sprint Goals

Based on the sprint planning described in Section 13.1 we decided on the following goals:

- Fill in the exercise function from document from the customer
- Implement the ability to view and create student pairs
- Add more filtering options to the exercise view
- Implement the ability to change profile picture in the app
- Implement a functional side menu which appears when tapping the icon on the navigation bar

This was almost our last sprint, so the focus here was to complete as much as we could. The report was always our main priority since it was important to document underway, through our development phase. We needed also to implement the other main feature for the application.

13.3. Sprint Backlog

The backlog for sprint 3 became like the Table 13.1.

Name	Estimated	Actual
View profile image and user name on home screen	4	5
Comment on scrapbook posts	8	16
Home screen tag statistics	5	7
Teacher pairs setup	50	30
Student pair viewing	20	11
Implement changing profile picture	30	37
Implement expandable side menu	10	13

Table 13.1.: Sprint 3's backlog: Estimation and actual effort are in hours.

13.4. Result from the Sprint

During this sprint we worked primarily on the features which allow the teacher to create pairs of students, assigning them to exercises and showing the pairs to students. We also updated previous components at the request of the customer. The statistics on the home screen had previously been placeholders.

13.4.1. GetToKnow

Viewing of pairs The implementation of the GetToKnow feature was divided into two parts: the student view and the teacher view. The student view can be seen in Figure 13.4.1. The teacher view was implemented as planned and assigns the students at the request of the teacher. The teacher view can be seen in Figure 13.4.1.

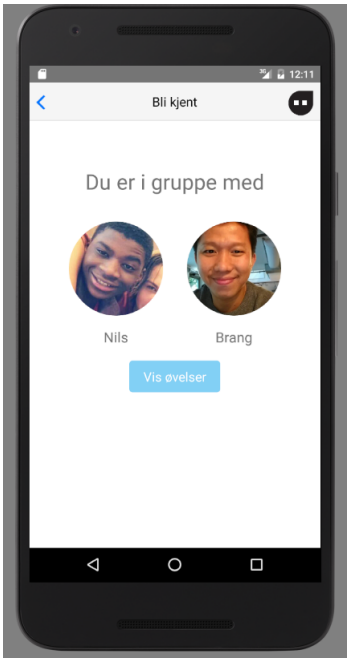


Figure 13.1.: Component for showing which pair the student is assigned to.

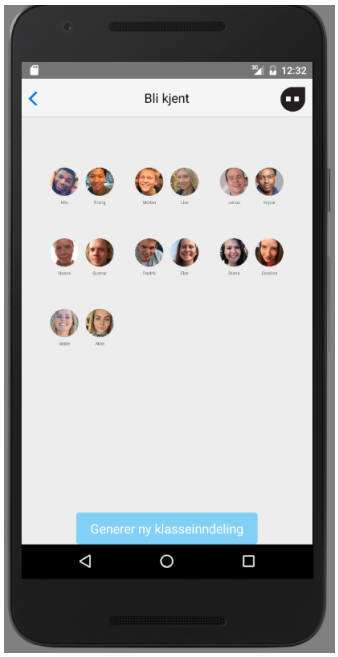


Figure 13.2.: Component for showing the current set of pairs and the ability to generate new pairs. This view is only accessible for the teacher.

Teacher Creation of Pairs The teacher can pair students into groups of two or three if there are an odd number of students. The teacher is given a preview of the pairs before approving the changes and after approving the pair layout they are required to select the exercises to choose from. The teacher can make use of the search-feature implemented in the last sprint to find fitting exercises for the relevant class. Currently the pairs are always the same, since we did not have time to implement an algorithm for deciding the best pair for each student.

13.4.2. Side Menu and Changing Profile Picture

The sidebar menu was implemented as a collapsable element that was shown when the user pressed the icon in the top-right corner of the navigation bar located at the top of the screen. The side menu includes the user's profile picture and name, some dummy tabs for notifications, and buttons for logout and changing profile picture. When the button for changing profile picture is pressed, the user is taken to a view that allows them to take a picture using their camera or choosing one from the library. The user is also given the option to crop their picture before uploading it to the database.

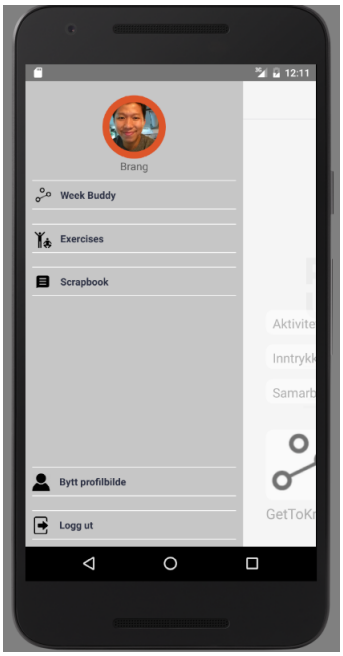


Figure 13.3.: Sidebar menu that is displayed when user presses the icon in the upper-right corner

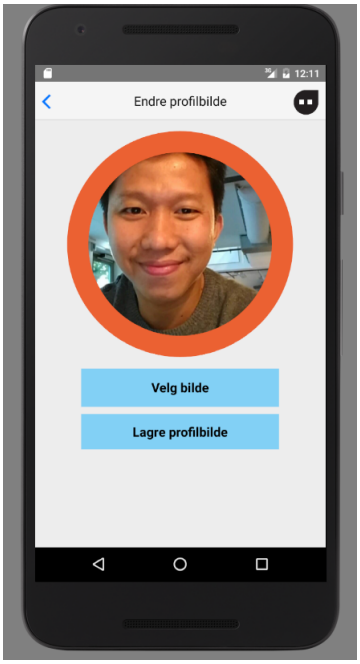


Figure 13.4.: Component showing the user a preview of their new profile picture, including button for selecting a new one and saving.

13.4.3. Scrapbook Comments

We managed to implement commenting on posts. Users can now write comments on posts that are instantly viewable by other users due to making use of the real-time database features of Firebase. Post comments are at first hidden to preserve space, however a button on each post allows users to show/hide the comment section on each post individually. The scrapbook with comments can be seen in Figure 13.4.3.

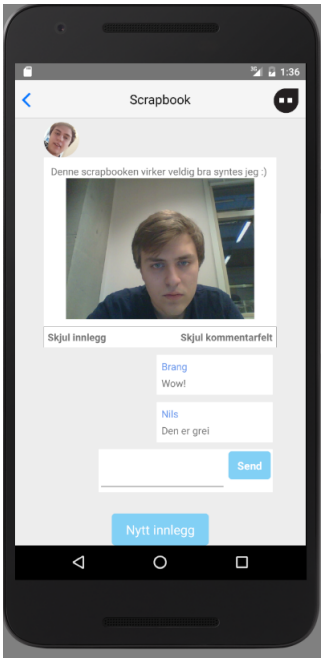


Figure 13.5.: An example post with the comment section visible

13.4.4. Home Screen

Home screen does now display a stats list that shows the basic tag statistics. The numbers next to the tags in the stats list shows the number of posts the tag is associated with. The statistics gets updated in real-time when a new post is uploaded.

We also managed to get rid of the static placeholder for the profile picture and user name. The image data and user information for the profile picture and user name is now fetched from Firebase.

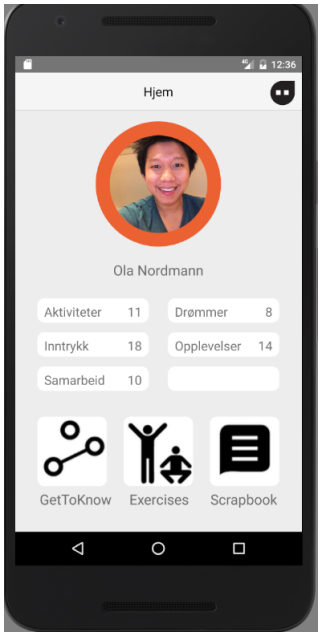


Figure 13.6.: Home screen showing scrapbook statistics and user information

13.4.5. Exercise List

After feedback on the list of exercises from last sprint we started working on a filtering system for the exercises. Together with the customer we decided that subject, themes, group sizing and estimated exercise time was good filters. We removed the tags previously added, and substituted them with, subjects, themes, grouping and time. A filtering system was then implemented to work alongside the search functionality implemented in the last sprint. It was created so that you can choose several filters from each category and exercises that matches all filters are shown in the list, while all along being searchable.

13.5. Customer Feedback

During the sprint review we got feedback from the customer relating the what was implemented during this phase. We agreed to discuss implementing some of the feedback in the next phase.

Exercise

The customer wanted that the student to get notifications when a teacher has added an exercise. But the questions such as when does this notification disappears was brought also up. They wanted also to make the view for both students and teachers different. The teachers should be able to plan and schedule exercises and make favorite buttons for

the exercises. The customer said that we should look at the planning image they posted on Facebook.

Scrapbook

The customer wanted the partner of the user to approve a post before the user posts the post. It should be possible to choose a category for the posts, which is already implemented but the customer wanted a set of standard tags. From this set of standard tags, the user can only choose those tags. One important thing that should be possible is to have a report button. When someone posts something inappropriate, other users should be able to report this with a report button. Or else, the most of the features in the scrapbook the customer liked.

13.6. Sprint Retrospective

At the end of the sprint we met internally in our group to summarize our progress during this phase. The meeting was shorter than our previous retrospect meetings, and there were less points raised than previously. We see this as an indication that we were getting into a good flow, and there was therefore less friction internally in the team.

What went well?

- We have gained enough knowledge about the development tools and frameworks to spend less time with unexpected issues than earlier sprints
- Had no issues communicating with customer like earlier.
- Using existing packages in our code where possible was a good idea.

What went wrong?

- Some merges were done without testing iOS support first, creating platform-specific bugs
- Some spelling mistakes and unorthodox coding styles were included in pull-requests

What can be done to improve?

- Make sure an iOS developer has reviewed the pull request before merging
- Developers should take a closer look at their code before submitting for pull request, making it easier to do code reviews before merging.

13.6.1. Sprint Burndown Chart

Figure 13.7 shown the burndown chart for this sprint. Effort remaining each day was calculated by estimating how much work was left on each task and summing the values.

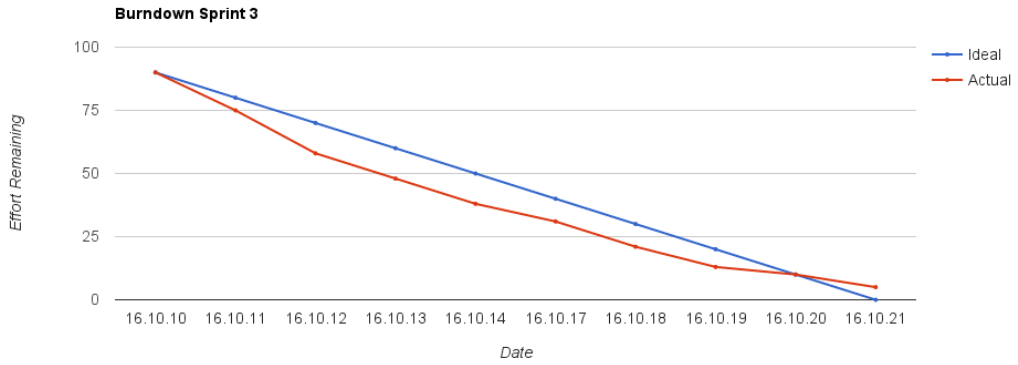


Figure 13.7.: Burndown chart for sprint 3

14. Fourth Scrum-sprint

14.1. Sprint Planning

We met with the customer to discuss what to work on during the final sprint. We had now finished most of the functionality we planned to implement, so it was important that we agreed on the final tasks that we would complete before ending development. We agreed that we would prioritize writing the documentation and looking over the code we had written earlier to make sure there were no remaining bugs or badly written code. There were also a few miscellaneous features still left to implement that we agreed with the customer that we would try to work on; The most important of these were support for multiple classes, algorithm for creating student pairs and handling of inappropriate scrapbook posts. We also had to perform usability testing with the prototype, and finalize the security of the application.

14.2. Sprint Goals

Based on the sprint planning described in Section 14.1 we decided on the following goals:

- Write documentation
- Refactor code where needed
- Fix any remaining bugs
- Add support for user to have multiple classes
- Prevent access to all exercises for students
- Add report button
- Add hiding of scrapbook posts
- Finalize report as much as possible
- Usability testing
- Finalize app security
- Create algorithm for optimizing pair creation

14.3. Sprint Backlog

We made the sprint backlog together with the customer at the sprint meeting, since this was the last sprint, it was very important to us and the customer that we agreed on what tasks we needed to finish before delivering the product to them. The tasks that were chosen are closely related to the sprint goals since the tasks are relatively small.

We also chose to spend time finalizing the security of the app, making sure that the security requirements specified in Section 9 are implemented to a satisfactory degree.The backlog for sprint 4 is shown in Figure 14.1.

Name	**Estimated**	**Actual**
Write customer documentation	10	7
Write ESdoc tags for all methods and classes	10	14
Add support for multiple classes per user	40	47
Add report button	2	2
Add hide post functionality	20	11
Write Firebase security rules	15	14
Prevent exercise list access for students	5	6
Design algorithm for optimizing pair creation	20	16
Implement algorithm for optimizing pair creation	40	46

Table 14.1.: Sprint 4's backlog: Estimation and actual effort are in hours.

14.4. Result from the Sprint

14.4.1. Implementing security rules

We finished implementing security rules for our Firebase backend during this sprint. Our access rules were based around class membership: Read access was restricted to students and teachers that had been given explicit access to that class.

Write rules depended on the type of object that were to be written; Exercises had no read access since they should only be changeable by an administrator. Scrapbook post write access however was granted to all students and teachers of that class, however they were designed to not allow users to overwrite posts they did not themselves create. Listing 14.1 shows an example of the implementation of Firebase security rules, which limits the access to Scrapbook posts.

Listing 14.1: Example Firebase security rules

```
"Scrapbooks": {
  "$class": {
    "posts": {
      "$post": {
        ".write": "(!data.exists() &&
        (root.child('class').child($class)
```

```
          .child('teacher').child(auth.uid).exists()
      || root.child('class').child($class)
          .child('student').child(auth.uid).exists()))
      || auth.uid === data.child('owner').val()"
    }
  },
  ".read": "root.child('class').child($class)
      .child('teacher').child(auth.uid).exists()
  || root.child('class').child($class)
      .child('student').child(auth.uid).exists()"}
```

14.4.2. Exercises

We made improvements to the exercise screen according to customer feedback. The new exercise list has a wider range of details to each exercise, each exercise now include keywords in the form of hashtags that give a quick summary of the exercise.

The Exercise View has been completely reworked with a new design as the product owner had provided sketches. A dummy rating system were also implemented as the customer wanted it for demonstration purposes. Figure 14.1 shows the new filter added in sprint 4 are visible, currently filtering exercises suitable for Norwegian class.

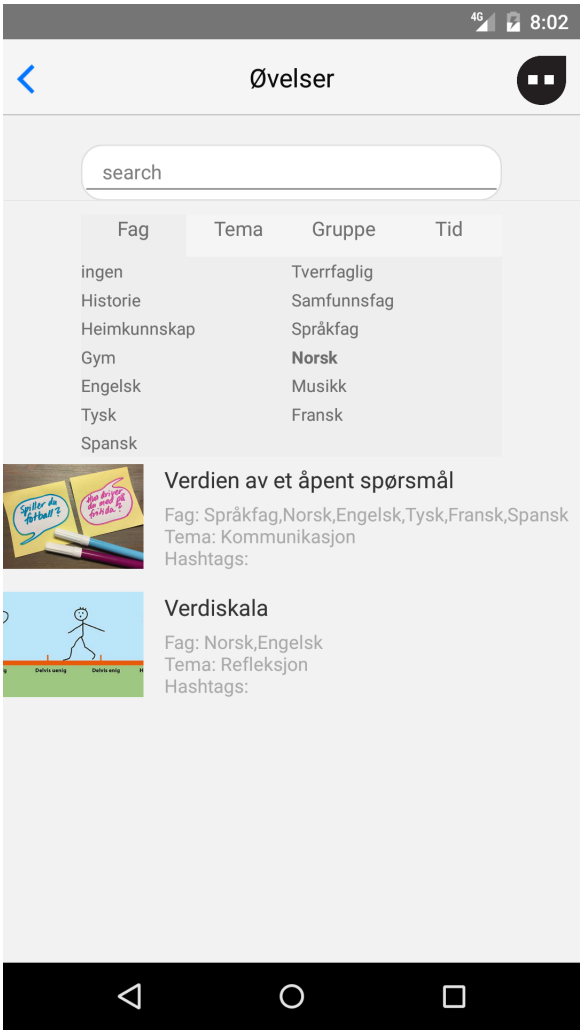


Figure 14.1.: New filters

14.4.3. GetToKnow Algorithm

In the previous sprint we did not implement any smart algorithm for distributing students into pairs. During this sprint we created an algorithm that tries to create pairs that have occurred few times earlier: The database keeps a record of previous groups, and the algorithm uses this to find the best pairs for each student. The algorithm is relatively simple and works by taking students without a pairs one by one from a list, then finding the student in the class that the first student has been on groups with the least; It repeats this until every student has a partner. If there is an odd number of students it will simply put that student on the group that was generated last.

14.4.4. Report and Hide Button

We implemented a simple report and hide button in the scrapbook. The report button is visible to only students while the hide button is visible to only teachers.

Report button (figure 14.2) allows students to report inappropriate behaviour to the teacher. Reported posts will be highlighted red in the teacher view.

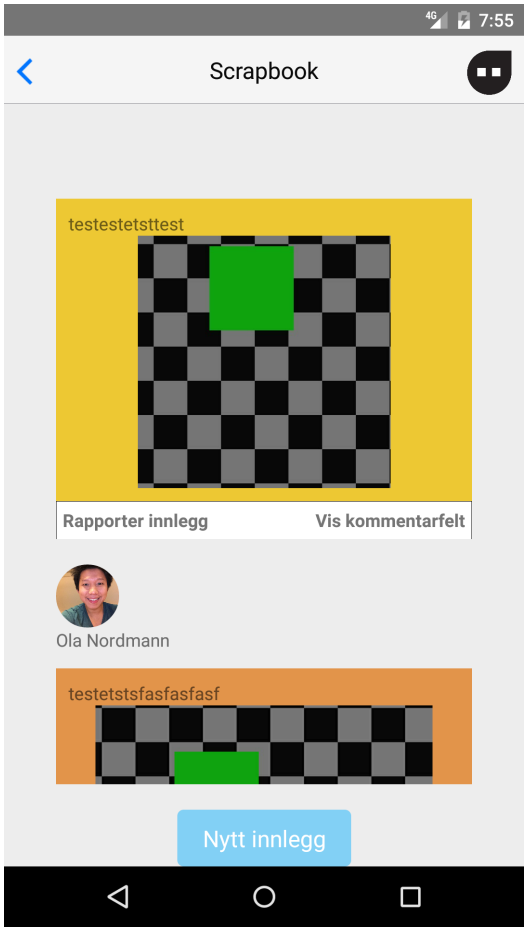


Figure 14.2.: Report button added in sprint 4 is visible on the Scrapbook.

Hide Button from figure 14.3 allows teachers to hide inappropriate posts. A post marked hidden is highlighted in blue and is invisible to students. Teachers can also unmark a post as reported by double tapping the hide button.

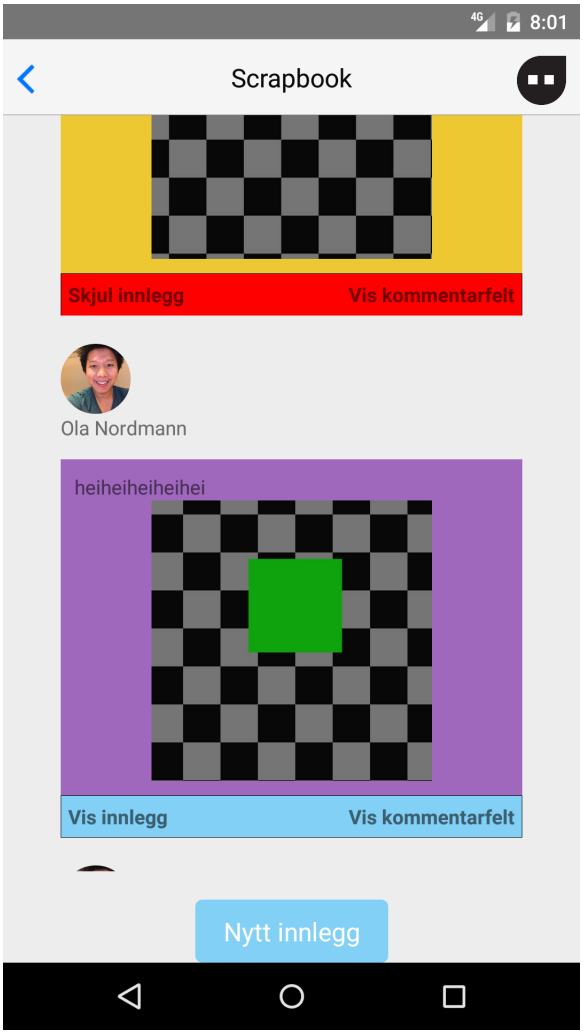


Figure 14.3.: Hide post button added in sprint 4 is visible on the Scrapbook.

14.4.5. Support for Multiple Classes

In earlier iterations of the prototype, there was only possible to log in to a single static class, even though the database supports multiple classes. We added multiple classes to the prototype by giving the user a choice of class after logging in based on what classes they are permitted to access in the database. If a user only has access to a single class, the option to choose is removed and they are sent directly to the home screen. This can be seen in figure 14.4.

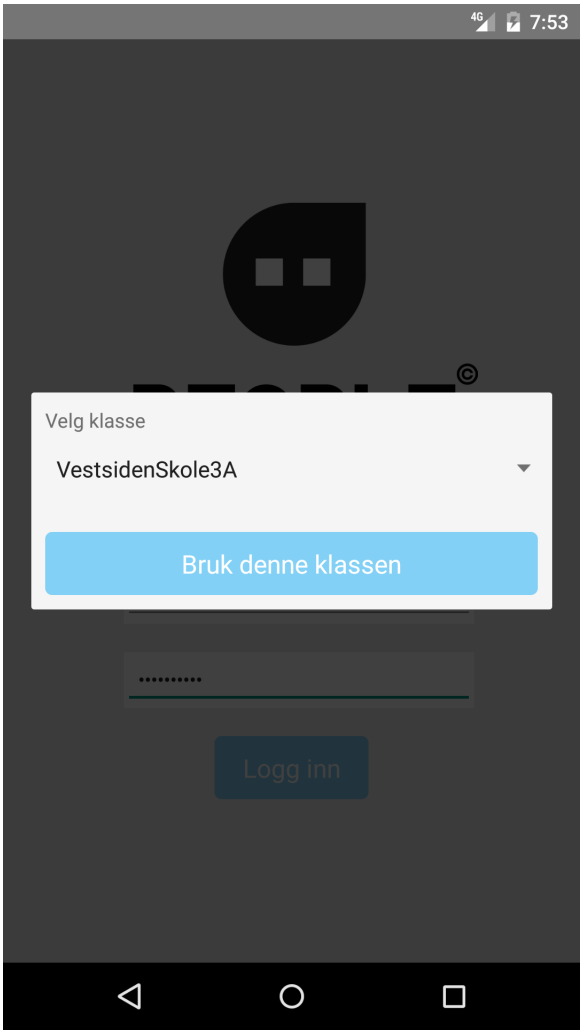


Figure 14.4.: User is prompted to pick the class they want to log in to.

14.5. Customer Feedback

We met with the customer after the sprint four ended to report back what we had worked on: we managed to finish everything we had agreed to do except for the functionality that allows teachers to remove absent students and change the who is paired together. While the customer had hoped that we would finish all goals specified during the sprint, they were overall happy with the changes we had made during the sprint. During the meeting, we got some feedback on the whole application. Because of this was the last sprint, we did not have time to add and modify the application from the feedback we got from the customer.

It was also understandable that our customer had more feedback since they are a startup, startup tends to have a lot of ideas and they are in a phase where they come up

with more ideas and input. All of the feedback we got from them was written down for the future, such that they can continue to improve the application. Some of the feedback is listed under:

Preventing Access to Exercise View for Students

We solved the problem with students having access to the full exercise list by removing the exercise button for students from the home screen. However the customer thought it would be better if the button was still there, but only showed the exercises assigned to the student by the teacher.

Reporting Posts

The report button we implemented allowed the students to report an inappropriate post, and the teacher could then hide the post from students. The post will then show up in the teacher's post and mark as hidden. But our customer wanted instead of this solutions, be able to collect all the reported post in one own view. This was because to have an overview so the teacher was aware that in this view contains only the reported posts.

Other Things

Other things we got feedback on was the algorithms that generate a group of three if the class contains an odd number of students. The customer wanted to make sure that a person that have been in a group of three should be next time in a group of two.They wanted also that the connection between pairs should be visible when one of them posted something on the scrapbook.

The student should also send an approval request to the partner when posting something such as exercises done together before posting on Scrapbook. The teacher should be able to modify pairs, students should be able to report to the teacher that they have completed the exercises, and teacher should be able to planning feature.

14.6. Sprint Retrospective

What went well?

- We managed to get more users for testing than expected
- Got started on development work early
- Did not introduce any bugs in the new features (That we know of)
- Manage to finish most of the requirements
- Got good feedback from the customer, and they were happy

What went Wrong?

- Didn't have time to finish one of the tasks in the backlog
- We had a bug in the application while doing usability testing, which broke the upload of pictures.
- Should be working more on the documentation since there still a lot to do.

What can be done to Improve?

- Have a backup plan under the usability testing
- Try to estimate tasks better to avoid dropped tasks
- Assign each member on focusing on the different tasks which are left
- Complete writing the documentation/ report

14.6.1. Sprint Burndown Chart

Figure 14.5 shown the burndown chart for this sprint. Effort remaining each day was calculated by estimating how much work was left on each task and summing the values.

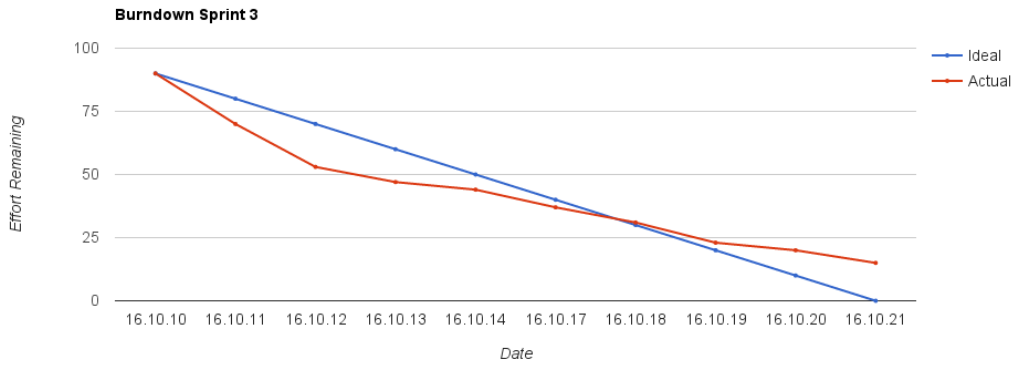


Figure 14.5.: Burndown chart for sprint 4

15. The Final Product

This chapter presents the finished product after all sprints we have been through. The chapter is organized in a way such that it presents each main feature, including pictures and the requirements. The requirements are from the table 5.1 in the chapter requirement analysis, and they have been shortened in order to make it readable.

15.1. Login

The Login view from Figure 15.1 is the first view in the application that appears first, when open PeopleUKnow application. The view consist of the logo of PeopleUKnow, email field, password field, and a log in button. When the user enters the password or the email wrong, a notification will pop up such as in Figure 15.2.

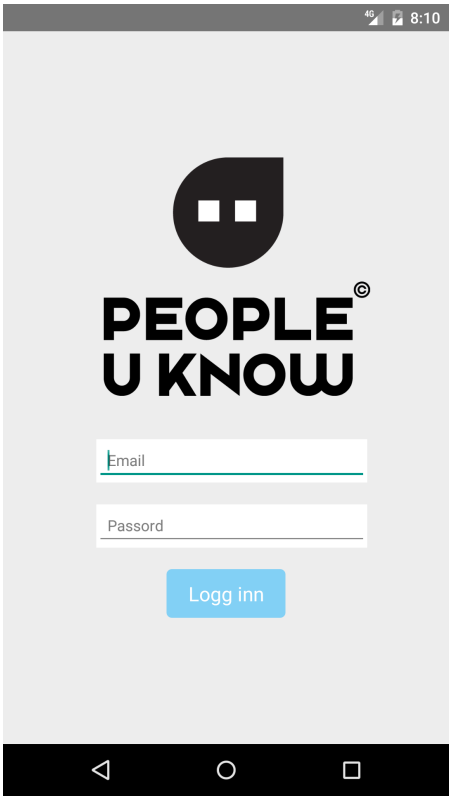


Figure 15.1.: Final Login View

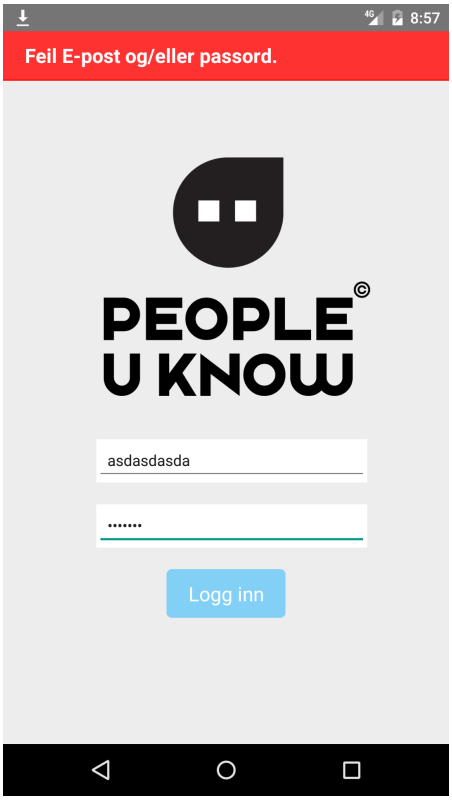


Figure 15.2.: Final Login View with "Sign-in failed" error

Completed Requirements

- F1 - Log in with email and password
- F2 - Notification when not successfully logged in

15.2. Home

The user will be presented with the home screen after the login view. From Figure 15.3, the user can see the profile picture and the name, and the user gets also an overview over the application; GetToKnow, Exercise, and Scrapbook. The user can also see class activities/ points on how many exercise the class has manage to complete together.

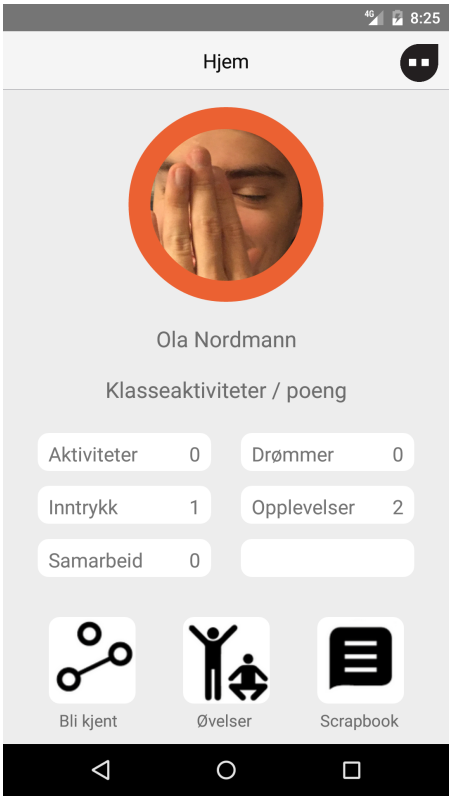


Figure 15.3.: Final Home View

The sidebar from Figure 15.4, gives also the user an overview of the application. It contains: the profile picture, week buddy, exercise, scrapbook, logout and change profile picture button.

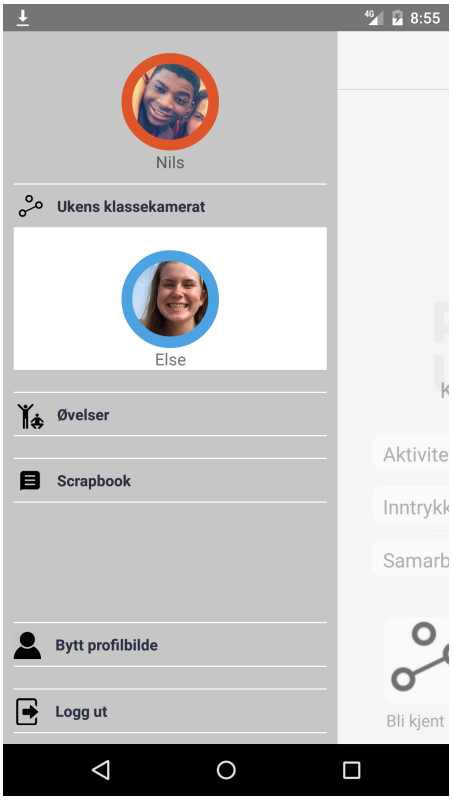


Figure 15.4.: Final Side Menu View

Completed Requirements

- F13 - Easy overview of the app features

15.3. GetToKnow

When the user press on the GetToKnow button from the Home view the user gets the view GetToKnow. This view shows the users partner (in the student view, Figure 15.5). The view display also the exercise button so the user can see the exercise the teacher has assigned to the group. For the teachers view (Figure 15.6), they can see the whole class and they can generate a new class with the generate class button.

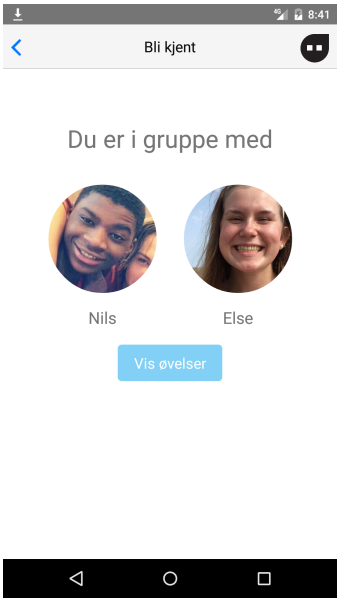


Figure 15.5.: Final GetToKnow Student View

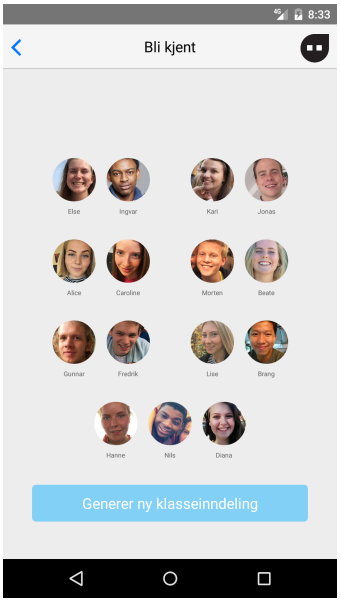


Figure 15.6.: Final GetToKnow Teacher View

Completed Requirements

- F6 - View weekbuddy

We manage to partly finish the requirement F7, which was to ”Group students into

pair and able to select who is attending and modify the pairs before my students can see the pairings". Some of the requirement F7 was not completed such as the teacher should be able to select who is attending and modifying of pairs.

15.4. Exercise

A user can choose Exercise feature from the Home view by selecting this button. Then depends on if the user is a student or a teacher the user will get different view. With the teacher's view, he/she can search on different exercise, and they will then get listed. When a user clicking on the picture, the user gets presented with a different view; description of the exercise.

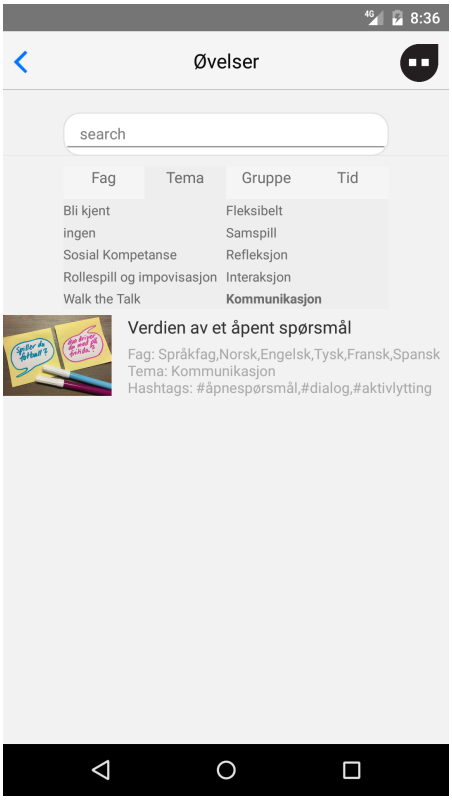


Figure 15.7.: Exercise View for Students

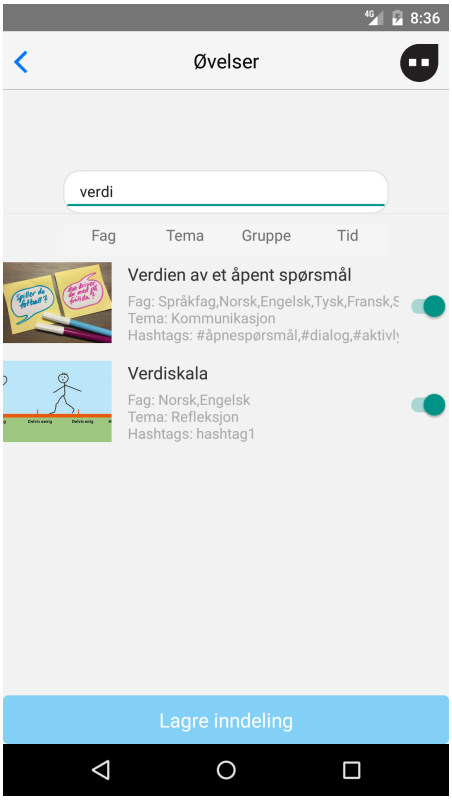


Figure 15.8.: Exercise View for Teachers

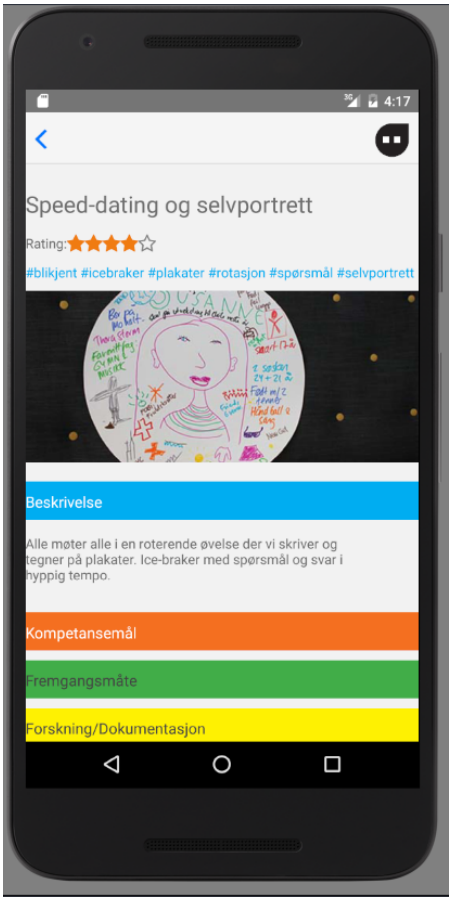


Figure 15.9.: Exercise View description

Completed Requirements

- F8 - View all the exercises which can be selected
- F9 - View Exercises assigned by the teacher

15.5. Scrapbook

The last feature provide the users with a Scrapbook. This is the class scrapbook which contains pictures and text posted by classmates from that specific class. A user can comment on the post by writing in the comment box. The user can also create a new post by pressing the ”nytt innlegg” button. Then the user will get a new view, where he/she can write something, then choose styling, add tags and a picture, and then post.

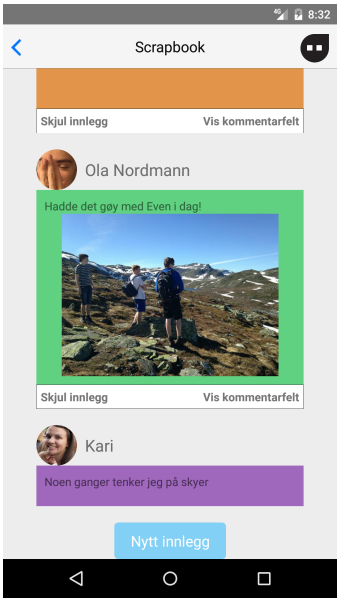


Figure 15.10.: Final Scrapbook

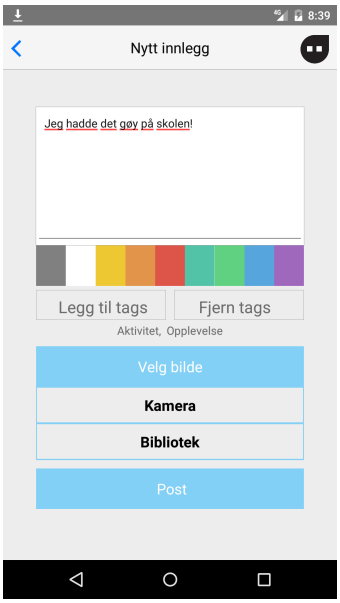


Figure 15.11.: Create Post in Scrapbook

If a post is inappropriate, a student can report the picture by pushing the report button in the scrapbook view shown in the Figure 15.10. Then a teacher can hide the post, with the hide function.

Completed Requirements

- F3 - View picture and text on the class scrapbook
- F4 - Comment on post on the Scrapbook
- F5 - Post pictures and text, and choose styling
- F10 - Search on the scrapbook on text and post-related data

16. Evaluation

In this chapter we describe our reflection on this project development. We will discuss group dynamics, what went well, what could be improved and further work.

16.1. Group Dynamics

This section describes our group dynamics such as what went well and what could have been done differently. The Figure 16.1 shows the group and our customers.

Figure 16.1.: The Customer and the Project Group

16.1.1. What Went Well

- Before we started the planning phase, we discussed what roles we should have and how many was necessary. Everyone was able to choose roles they were interested in, and each team member was able to work on something they wanted to. The only role we had no choice in was team leader, which was predetermined. Even though we had specific roles, we delegated responsibility on tasks that were too large to handle alone, this meant that everyone got to try out roles they were not directly responsible for.

- The group was good at helping each other and if there were problems we felt it was easy to ask each other for help. Since everyone had different levels of experience with coding it was important that everyone felt comfortable asking others for help. We felt we saved a lot of time by not hesitating to help each other out.

- Our meetings with the customer and within the group were effective. Maybe the first time it took a lot of time, but this was the first time we all met and a lot of things needed to be clear and therefore it took much time. But we learned from this and made templates for meeting and planned for each meeting on what to discuss and what to talk about on the meeting.

- The choice of agile development proved to be a good choice; Since we worked closely with a customer on changing requirements it was very important that we communicated regularly and efficiently during all phases of the project. If we had chosen a method like waterfall where everything is done in rigid phases, we would not have been able to work as closely with the customer throughout the project.

16.1.2. What Could be Improved

A group project is never completely perfect. It can always be improved. In our group there were things that could have been done differently:

- Communication tools; we should have used slack in the beginning because we started out to use Facebook. This made us used to used it to communicate with each other, while slack was introduced later. Slack was a great tool that we could have been using more because it was easy to go back and find the information which was said.

- The group uses a lot of time on meetings, maybe we should send one person from the group instead of the whole group. Instead of being in the meeting, they could have continue working on the project, and maybe have a meeting within the group where the group leader or scrum master updated the rest of the group.

- Booking of rooms should have been done earlier. We manage to book rooms for eight weeks, but some of the room we got were not the bests ones. Since some of them had few electrical socket, which was a problem when we all needed electricity to our computers.

16.2. What We have Learned

From this project, we got to learn a lot of different things not only about technologies but different methods and working with groups. We also learned from each other. Even though we all study computer science, we all have difference experience and different backgrounds and different qualities.

Cross-platform app development

None of us had any real experience with cross-platform app development at the start of the project, and got a lot of real experience with this by using React Native. If we had opted for developing the app natively on iOS and Android we would effectively have had to do over twice the amount of work for the same result. We also believe the customer will greatly benefit from this choice in the future; If they want to improve on our product and bring it to market quickly they will benefit from the decreased amount of work cross-platform app development brings.

Working with a Real Customer

While we have taken courses earlier that simulated working with a customer; We had never worked with a real one before. We quickly learned that working with a real customer with a different professional background had its challenges but also brought a lot of benefits. We learned that making use of the customer's expertise can be greatly beneficial to the outcome of the project, as they might have knowledge relevant to the project we do not.

When working with a real customer we also learned that it is vital that both the team and the customer have the same expectations of what is produced. We learned that differences in technical experience can make expectations differ between parties involved in the project, and it is therefore vital that these differences are removed before any real work can be started.

The importance of planning and coordination of work

Working in a team of six people we quickly learned it was important to plan ahead and make sure everyone knew what they were supposed to do. We experienced that when we didn't plan work to a sufficient degree we had more disagreements and problems than when we were careful with planning. We also learned that when work actually began it was important to keep each other up to date on what was worked on, even when not working on the same thing; This is because it is often possible to reuse work done on one feature in others. When we didn't coordinate properly during work we often ended up doing twice the amount of work and having to scrap some of it.

16.3. What the Future Customer-Driven course Student should know about the course

Get to know your team-members immediately We waited until after the first customer meeting before we had an internal meeting with only the team members. If we were to do this again we would have met internally before the first customer meeting so that we all agreed on what we wanted to get out of this course, and thus be better prepared to discuss our project goals during the customer meeting.

Transfer experience between team members early Each team member had different levels of experience with software development and all had qualification the others did not. Instead of transferring skills when needed in an unorganized fashion, we should have identified who had important experience, and had them transfer this experience to the other team members early in the project. This could lead to less friction later in the project due to different skill-levels, and overall increase the effectiveness of the team.

16.4. Feedback on the Customer Driven course TDT4290

- The course was very interesting because it gave us experience with working with a real customer.
- Our supervisor was really helpful and gave us good feedback on our report and made sure that we were on the right track.
- Many of the mandatory presentations were interesting and relevant to the course.
- The course could have done a better job on keeping groups up to date with the latest changes or notifications. The website that was used was not an efficient way of distributing updates. We were often not notified of changes to the course until after the weekly supervisor meeting.
- There could have been more information about the course given out when it started: We thought that we would be given a better introduction to the course on the first day, the first customer meeting came on too suddenly since we had barely had time to meet those we would be working with.
- There was a lot of confusion at the start of the project since the compendium was not updated until a few weeks after the course had started. Since the old compendium contained conflicting information with the new one, it caused some confusion internally that impacted project negatively. In the future the new compendium should be available when the course starts.

16.5. Further Work

During our project we chose to focus on the core-functionality of the app. Since the customer is still in the early phases of development, they still have many ideas that they want to add to the prototype we made. In this section we have detailed some of the features that we either didn't have time to do, or were brought up but never made it to the product backlog.

Feide integration Since the customer wants to eventually sell their product to schools, integration with the Feide database was a feature that the customer felt was very important. Feide integration could simplify registration of students by retrieving student lists directly from Feide and allowing students and teachers to log in to the app using their Feide account. It could also be possible to retrieve useful data from Feide or other data-stored associated with Feide, for instance automatic detection of absent students in the class.

More teacher control over pairing process Due to conflicts between members of a class it could be undesirable to pair certain students. It could also be good for specific students to be paired together if the teacher thinks they could benefit from doing exercises together. The prototype we made only pairs students based on how often they have been in pairs earlier, and the teacher cannot override the choice made by the app. Implementing functionality that allows the teacher to override the pair choices and to create better pairings for that specific class could make the app more useful.

Handling of absent students In the prototype we made there is no handling of students that are absent from the class. When a student is absent it can mean that one student is left without someone to work with on exercises. Instead of having teachers manually resolve this problem it could be possible for the app to register which students are absent, and take the appropriate measures to make sure no student is left without a partner.

Creating long-term exercise plans In the prototype we made, each class can only have a single set of exercises active at any time; This could be problematic for teachers as they cannot plan new exercises without overwriting the old ones. In the future it could be possible for teachers to assign certain exercises to specific time-slots during the week, allowing teachers to plan ahead to much higher degree.

Pair-related scrapbook post The scrapbook is a place where students are supposed to share experiences together. Currently each scrapbook post is only bound to a single student, however it would be convenient for students to tag their posts with who else in the class took part in the making of the post. This would also require a system where students approve being tagged in a post, so that a student cannot be tagged in a post without consent.

Searching on the scrapbook The scrapbook currently works in much the same way as a normal feed: Posts are listed in chronological order and finding an old post can therefore be difficult. It could be convenient for users to have the ability to search for specific text or metadata to find old posts more easily.

Lunch date The ability for students to sign up for a lunch date with other students was discussed early in the project but was never added to any sprint. This feature would allow students to be assigned someone to spend their lunch break with automatically.

16.6. Conclusion

Working on a real software development project with PeopleUknow has been a learning experience. On one side, it was exciting and interesting, but on the other, it was also stressfull and timeconsuming. None of the members on our team had worked on a project as big as this before, but we enjoyed it and felt that developing a software system is far more motivating when a real customer is involved. The goal was to create a mobile application prototype that could function as, or be extended to work as an MVP for PeopleUknow. After our final acceptance test we agreed that even though there is always room for improvement, the project was a success.

A. Appendices

A.1. Pre-Study

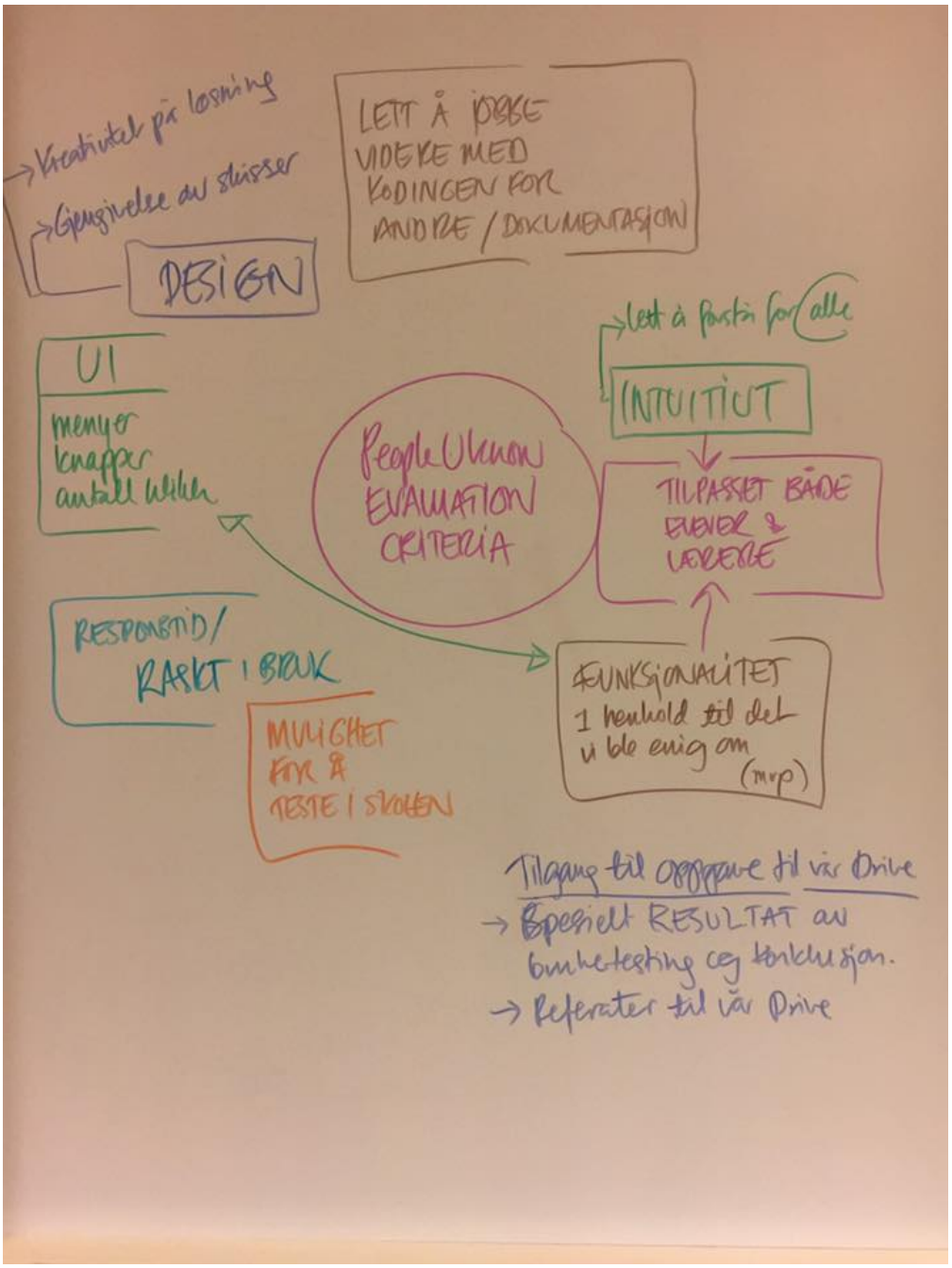


Figure A.1.: Evaluation Criteria

A.2. Risk Tables

Risk id	1
Activity	All
Risk factor	Members of the group may get sick, or are not showing up
Impact	M: This can reduce the quality of the product
Probability	M: It is easy to get sick when other is sick
Strategy and action	Avoid: Each member has to take good care of themselves. Be clear about how important that everyone is showing up. Everyone need to inform/alert group member why they didn't show up
Responsibility	Project Manager

Table A.1.: Risk 1

Risk id	2
Activity	All
Risk Factor	Things/materials get stolen
Impact	M: Impact on the group member's productivity
Probability	L: This can happen but we all know how to watch our stuff
Strategy and Action	Avoid: Don't put your things everywhere, take good care of them, remember to take backup
Responsibility	Each group member

Table A.2.: Risk 2

Risk id	3
Activity	All
Risk Factor	Conflicts in the group
Impact	M: Can reduce the efficiency of the group
Probability	H: Since we have different experience, knowledge and some can have strong opinions
Strategy and Action	Avoid: Be open, discuss the problem, see the problem from different perspective, set a deadline for when to end the discussion, and if it is necessary get some assistance from supervisor
Responsibility	Project manager

Table A.3.: Risk 3

Risk id	4
Activity	Report and planning
Risk Factor	Loss of data on Google Drive or Sharelatex
Impact	H: Have to rewrite report, permanent loss of documents
Probability	L: since Google drive and sharelatex are used by many people, they have to manage to deliver their services
Strategy and Action	Avoid: Backup data locally
Responsibility	Report manager

Table A.4.: Risk 4

Risk id	5
Activity	Planning and development
Risk Factor	Unable to find work place
Impact	L: Need to work from home. Reduces the communication in the group.
Probability	M: Many students are early with booking rooms
Strategy and Action	Avoid: Book rooms in advance
Responsibility	Project manager

Table A.5.: Risk 5

Risk id	6
Activity	Product development
Risk Factor	Underestimated the time to complete tasks
Impact	H: Some features have to be dropped and/or team have to work overtime
Probability	M
Strategy and Action	Avoid: Add slack to tasks in planning phase so we have a buffer in case of underestimation
Responsibility	Project manager

Table A.6.: Risk 6

Risk id	7
Activity	Development of the product
Risk Factor	Unforeseen technical issues
Impact	H: Code has to be rewritten
Probability	M: Since we are working with many different technologies
Strategy and Action	Reduce: Make sure developers have a good understanding of the technology in relation to the use-cases
Responsibility	System architecture

Table A.7.: Risk 7

Risk id	8
Activity	All
Risk Factor	Communication problems
Impact	M: Impact on the productivity to the group and the quality of the product
Probability	M: We communicate with each other every time we meet, so it's possible that this can happen
Strategy and Action	Avoid: Be clear when communicating, ask questions if you don't understand, go through the task together to ensure everyone understand
Responsibility	Project manager

Table A.8.: Risk 8

Risk id	9
Activity	Developement and report
Risk Factor	Tool risk: When using a new too
Impact	M: Can take a big amount of time because we need to understand how the tool is working
Probability	M: We use many tools for this project, but some of them we are familiar with
Strategy and Action	Avoid: Try to use the tools that we understand. If it is necessary to use other tools, try to ask someone or google it
Responsibility	Project manager

Table A.9.: Risk 9

Risk id	10
Activity	All
Risk Factor	Lack of knowledge
Impact	H: Impact on the performance and the quality of the product
Probability	M: We have enough knowledge to take this course, but we may need knowledge about the customers product and technologies they are using.
Strategy and Action	Avoid: Learn more on that specific subject Reduce: Work on the topic the member have knowledge about.
Responsibility	The person with the most knowledge in that area.

Table A.10.: Risk 10

Risk id	11
Activity	All
Risk Factor	Wrong choice(s) have/has been made in the middle of the process
Impact	M: Impact on the time
Probability	M: We all have some experience with taking choice etc. but sometimes we don't have enough knowledge to do the right decision
Strategy and Action	Reduce: Go through all the choice the group have and assess which one is better of Accept: yes, the group has made some wrong choices, but then moves on and learn from it. If necessary start from the beginning
Responsibility	

Table A.11.: Risk 11

Risk id	12
Activity	All
Risk Factor	The customer unavailable
Impact	H: Impact on the quality of the product.
Probability	H: Our customer is busy with alot of stuff, and sometimes the customer is gone for a business trip
Strategy and Action	Avoid: schedule meeting with the customer.
Responsibility	Customer Relationship Manager

Table A.12.: Risk 12

Risk id	13
Activity	All
Risk Factor	A project member drops the course.
Impact	H: Loss of productivity
Probability	L: This is a mandatory course for 4th years student
Strategy and Action	Avoid:Make sure everyone is happy with project progress, conflicts are resolved quickly and notify early if a member wants to drop the course.
Responsibility	Project Manager

Table A.13.: Risk 13

Risk id	14
Activity	All
Risk Factor	Misunderstanding between the group and the customer
Impact	H: Design/code something the customer did not want.
Probability	H: Customer have a different background than us and maybe wants other features
Strategy and Action	Avoid: Set up regularly meetings with the customer and agree on the desgin
Responsibility	Customer relationship manager

captionRisk 14

Risk id	15
Activity	All
Risk Factor	Low motivation
Impact	M: Impact on the quality of the product
Probability	M: sometimes the low motivation can suddenly kicks in, we are all human
Strategy and Action	Avoid: Include everyone in the team weekly updates and meeting, assign members with tasks. Members should be honest if they are lacking of motivation.
Responsibility	Project manager

Table A.14.: Risk 15

A.3. Use Case

Use Case #	2
Application Name	PeopleUKnow App View scrapbook
Requirements	F3
Description	The user wants to view post from other student in the class.
Primary Actors	Students
Preconditions	• The user is logged in • The user has a class
Trigger	The user wants to see what has happened in his/her class on the scrapbook.
Basic flow	• The user is presented with a menu bar. • The user navigates to the scrapbook. • The user views the scrapbook • The user swipe and scroll down/Up on the Scrapbook
Alternative	
Post conditions	The user has viewed the scrapbook.

Table A.15.: Use case 2

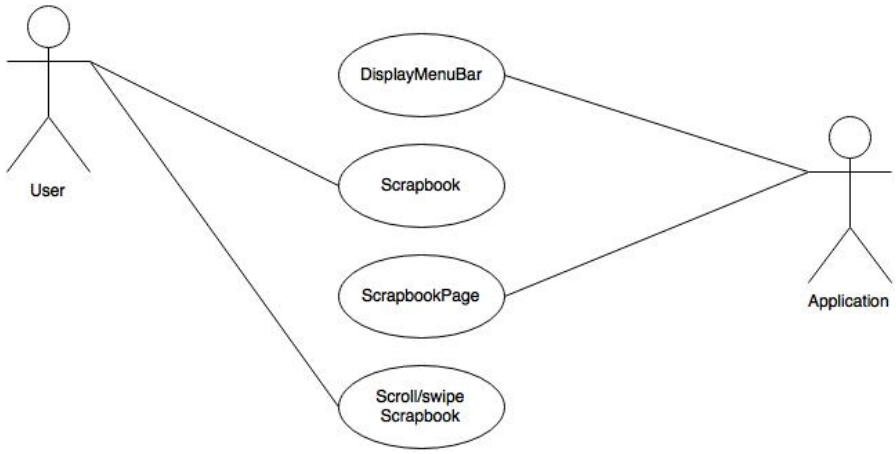


Figure A.2.: Use Case Diagram 2: View Scrapbook

Use Case #	3
Application	PeopleUKnow App
Name	Comment on Post
Requirements	F4
Description	The user wants to comment on post made by his/her classmate on Scrapbook
Primary Actors	Students
Preconditions	• The user is logged in • The user has a class • There exists some post in the scrapbook
Trigger	The user has seen something interesting she/he wants to comment on.
Basic flow	• The user see a post made by a classmate • The user write in the comment box • The user see his/her comment
Alternative	
Post conditions	The user has comment on the post.

Table A.16.: Use case 3

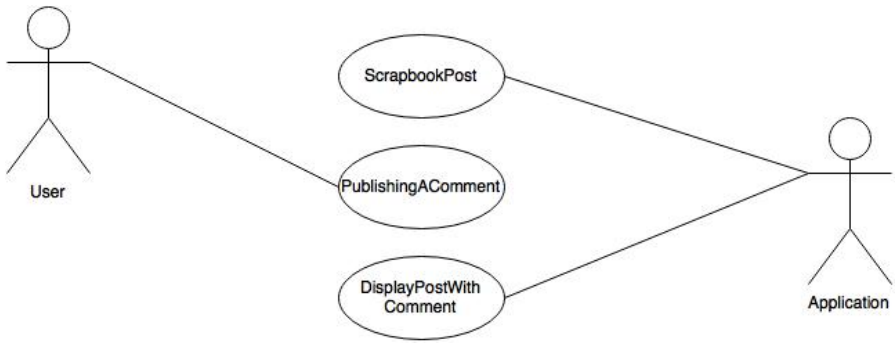


Figure A.3.: Use Case Diagram 3: Comment on Post

Use Case #	5
Application	PeopleUKnow App
Name	Viewing Pairs
Requirements	F6
Description	The student wants to view which pair he/she has been assigned by the teacher
Primary Actors	Students
Preconditions	• The user is logged in • The user has a class
Trigger	The user has been assigned a partner.
Basic flow	• The user choose the "GetToKnow" function from the menu bar • The user is presented with all the connection of pairs in class • The user see her/his partner(s)
Alternative	
Post conditions	The user view his/her partner(s)

Table A.17.: Use case 5

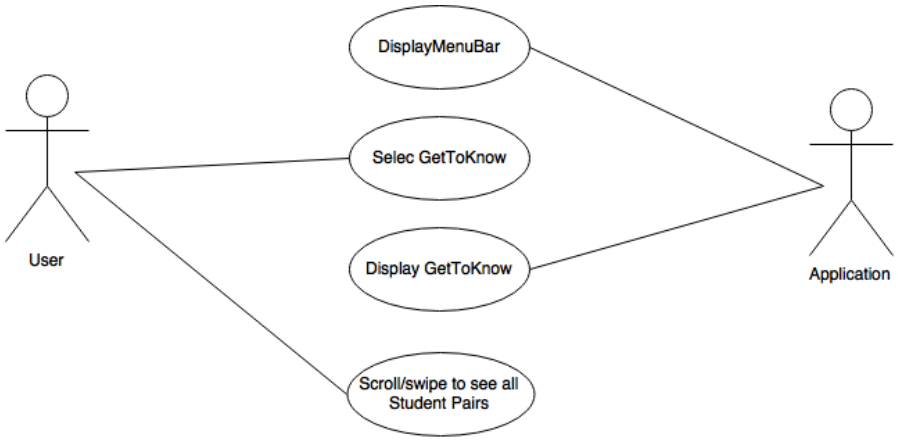


Figure A.4.: Use Case Diagram 5: View the GetToKnow pairs

Use Case #	7
Application	PeopleUKnow App
Name	Teacher Views and Choose Exercises
Requirements	F8
Description	The user wants to view all the exercises that he/she is able to select for the student pairs.
Primary Actors	Teachers
Preconditions	• The user is logged in • The user has a class
Trigger	
Basic flow	• The user choose “exercise” in the menu bar • The user is presented with different type of categories • The user choose to view all • The user scroll down to see all the exercise
Alternative	
Post conditions	The user can view all the exercises.

Table A.18.: Use case 7

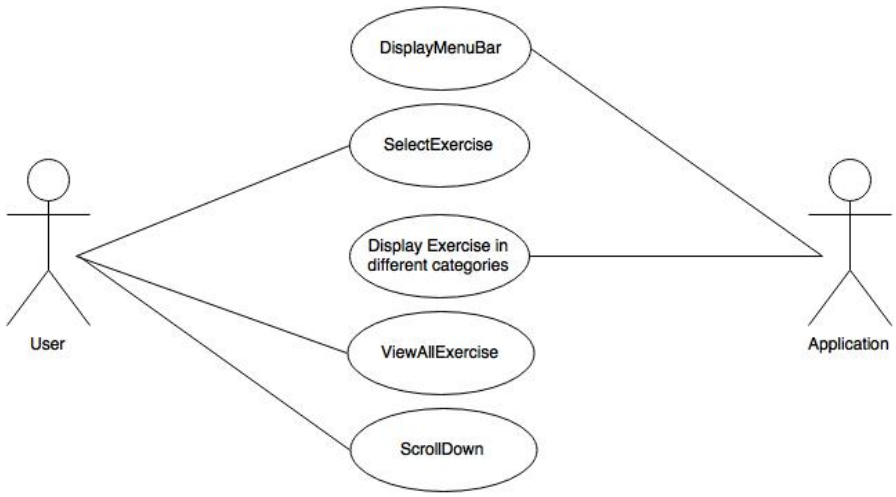


Figure A.5.: Use Case Diagram 7: Teacher Views and Choose Exercises

Use Case #	9
Application	PeopleUKnow App
Name	Search on Post-related Data
Requirements	F10
Description	The user wants to find a post related to a category by searching on post-related data.
Primary Actors	Students, Teachers
Preconditions	• The user is logged in • There has been posted something on the scrapbook
Trigger	The users wants to check a theme on the scrapbook
Basic flow	• The user navigates to “Scrapbook” from the menu bar • The user is presented with the post on the scrapbook • The user choosing the search function • The user enter word(s) in the search field • The user is presented post related to that search
Alternative	
Post conditions	The user has manage to search on a category on the scrapbook

Table A.19.: Use case 9

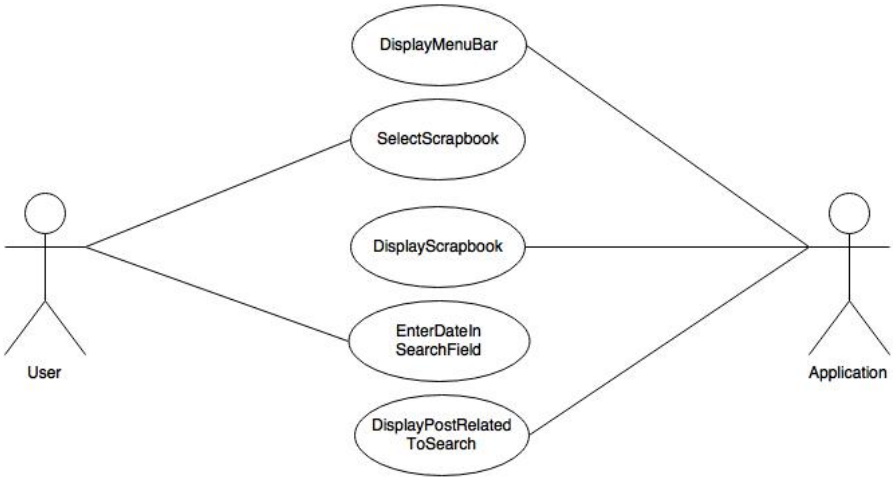


Figure A.6.: Use Case Diagram 9: Search on Post-related Data

Use Case #	10
Application	PeopleUKnow App
Name	Sign Up for Lunch-date
Requirements	F11
Description	The user wants to sign up for a lunch-date with another student.
Primary Actors	Students
Preconditions	• The user is logged in • The user has a class
Trigger	The user wants to eat with another classmate
Basic flow	• The user choose “lunch-date” from the menu bar • The user schedule when the user wants to eat • The user is presented with the preferred time for the user
Alternative	
Post conditions	The user has successfully been signed up for a lunch-date

Table A.20.: Use case 10

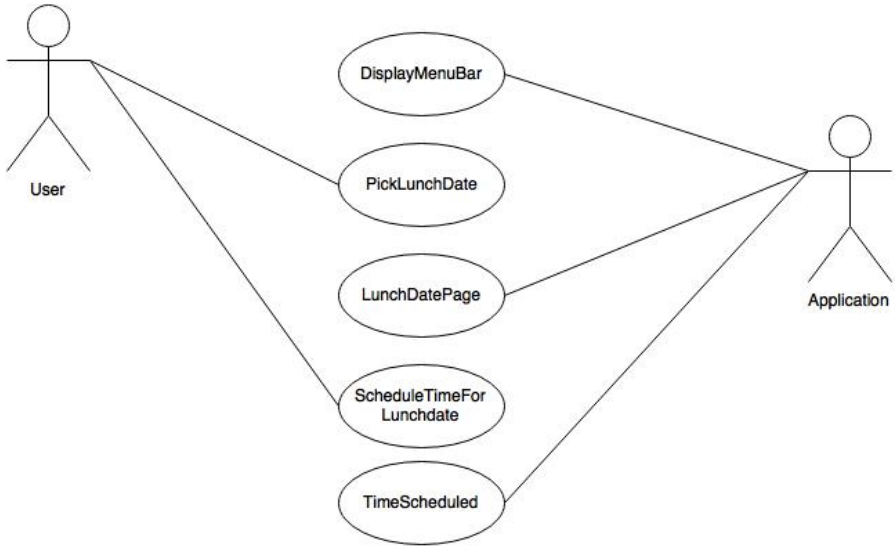


Figure A.7.: Use Case Diagram 2: Sign Up for Lunch-date

Use Case #	11
Application	PeopleUKnow App
Name	Paired Up for Lunch-date
Requirements	F12
Description	The user gets an notification when the student gets a pair for lunch-date and can view the lunch-partner.
Primary Actors	Students
Preconditions	• The user is logged in • The user has signed up for lunch-date • Another user has signed up for lunch-date
Trigger	
Basic flow	• The user gets a notification from the application • The user opens up the notification • The is presented with a lunch-partner
Alternative	
Post conditions	The user has been notified when the user got a lunch-date and the user can see with who.

Table A.21.: Use case 11

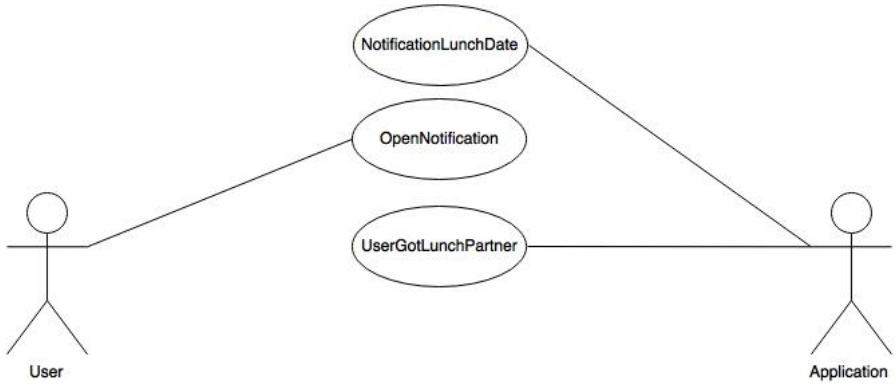


Figure A.8.: Use Case Diagram 11: Paired for Lunch-date

Use Case #	12
Application	PeopleUKnow App
Name	Overview of Application Features
Requirements	F13
Description	The user wants to get an overview over all the features the application has and be able to open the pages for the features.
Primary Actors	Students, Teachers and Parents
Preconditions	The user is logged in
Trigger	The user wants to get to know the application.
Basic flow	• The user is presented with the menu bar • The user push each button on the bar • The user gets to the next page for the feature/button
Alternative	
Post conditions	The user has got an overview over the whole application and got into each features

Table A.22.: Use case 12

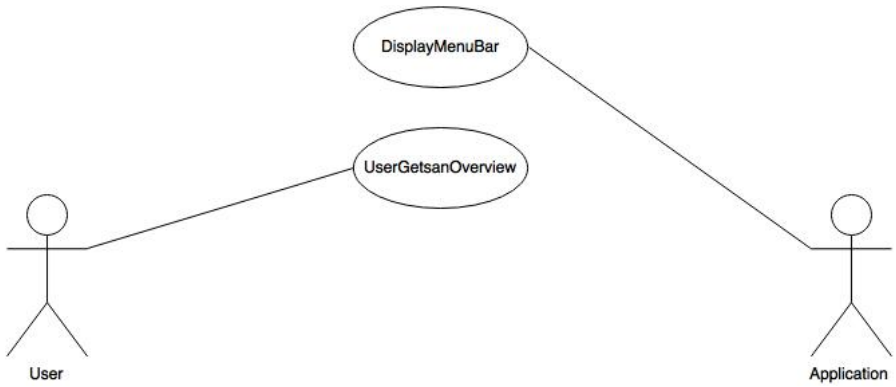


Figure A.9.: Use Case Diagram 12: Overview of the Application

B. Planning

B.1. Meeting templates

B.1.1. Meeting within the Development Team

```
1.Approval of agenda
2.Approval of minutes of meeting from last advisor meeting
3.Comments to the minutes from last customer meeting or other meetings
4.Approval of the status report, which may be structured as follows:
4.1 Summary
4.2 Work done in this period Status of the documents that are being created
Meetings Other activities
4.3 Problems { what is interfering with the progress or taking resources?
Problems are often risks that have taken effect.
4.4 Planning of work for the next period Meetings Activities
4.5 Other
5.Review/approval of attached phase documents
6.Other issues are listed here...
7.Other issues
```

B.1.2. Daily Stands-ups

```
Done last time:
...
Going to do today:
...
```

B.1.3. Customer Meeting

```
- Show what has been done
- Discuss goals
- Customer feedback
- Agree on features for next increment
- Plan next meeting
```

B.1.4. Supervisor Meeting

```
- Review report
- Questions
- (Demo of product)
```

B.1.5. Weekly Report

```
- Deliver report so far
- What had been done?
- What we should do?
- Questions about the report for supervisor for next meeting?
- Questions about the course for supervisor for next meeting?
```

C. Software Security

C.1. Protection Poker

Feature	Exposure Value	Assets	Asset value	Overall score (AV + EV)
F1	100	User info Credentials Firebase kildekode database	340	34000
F2	<10	User credentials Database Firebase Kildekode	280	2800
F3	50	Bilde/post User info Firebase Kildekode Database Exercise	360	18000
F4	60	Bilde / Post Db kildekode	190	11400
F5 -	80	Bilder/ Post Db YOutube User credentials kildekode	290	23200
F6	30	Bruker info Database Firebase Bilder Kildekode	350	10500
F7	60	Bruker info Database Firebase Bilde Kildekode	350	21000
F8	50	Exercise Youtube Database Firebase Kildekode	250	12500
F9	30	Exercise Youtube	250	7500

		Database Firebase kildekode		
F10	30	Bilde/post user info Database Firebase kildekode	350	10500
F11	50	Bruker info bilder database firebase kildekode	350	17500
F12	40	Bruker info bilder database firebase kildekode	350	14000
F13	<10	Bilder/post Kildekode	110	1100

D. Testing

D.1. Functional Testing

D.1.1. Functional Testing, sprint 1

Functional Test 1	
ID	FT1
Description	Tries to log in
Date	27.09.16.
Tester	Ivar
Preconditions	App must be running, password and email correspond, and exist in the database
Tasks	1. Type in email 2. Type in password 3. press the log in button 4. confirm that you are redirected to home-screen
Result	When the log in button is pressed the user is redirected to the home screen.

Table D.1.: Functional Test 1

Functional Test 2	
ID	FT2
Description	Validate the possibility to navigate to components from home screen
Date	29.09.16.
Tester	Ivar
Preconditions	App must be running and a user must be logged in on the home screen
Tasks	1. Press the exercises button 2. Validate that app shows the exercise view 3. press the back button 4. confirm that you are redirected to home-screen 5. Press the getToKnow button 6. Validate that app shows the getToKnow view 7. press the back button 8. confirm that you are redirected to home-screen 9. Press the scrapbook button 10. Validate that app shows the scrapbook view 11. press the back button 12. confirm that you are redirected to home-screen
Result	It is possible to navigate back and forth between the home screen and the different functionality

Table D.2.: Functional Test 2

D.1.2. Functional Testing, sprint 2

Functional Test 3	
ID	F2.1
Description	Tries to post image on scrapbook
Date	7.10.16.
Tester	Matias
Preconditions	App must be running, and user must be logged in
Tasks	1. Press Scrapbook button 2. Press new Post button 3. Add text 4. Add at least one tag 5. Press add image, and select from library 6. Select an image. 7. Press Post button
Result	When the Post button is pressed the user is redirected to the home screen, and that the post was made can be verified on Firebase.

Table D.3.: Functional Test 3

Functional Test 4	
ID	FT2.2
Description	Tries to view specific exercises
Date	7.10.16.
Tester	Matias
Preconditions	App must be running, and user must be logged in
Tasks	1. Press exercise button 2. Scroll until you see desired exercise 3. Press exercise 4. Read info regarding exercise
Result	When the exercise is pressed, an image from the exercise is displayed along with text that explains how the exercise is done.

Table D.4.: Functional Test 4

D.1.3. Functional Testing, sprint 3

Functional Test 5	
ID	FT3.1
Description	Tries to change profile picture
Date	20.10.16.
Tester	Vigleik
Preconditions	App must be running, and user must be logged in
Tasks	1. Press sidemenu button 2. Press "change profile picture" 3. Choose picture from gallery or camera 4. Press "save profile picture"
Result	When the save button is pressed the user is redirected to the home screen, and the new profile picture is uploaded to Firebase. The old profile picture displayed on the home screen is automatically refreshed with the new one

Table D.5.: Functional Test 5

Functional Test 6	
ID	F3.2
Description	Tries to leave a comment on a post
Date	20.10.16.
Tester	Vigleik
Preconditions	App must be running, and user must be logged in. There must exist a post
Tasks	1. Press scrapbook button 2. Press "show comments" on an arbitrary post 3. Type in a comment 4. Press send
Result	When the send button is pressed the comment will be uploaded to Firebase and automatically updated on the app.

Table D.6.: Functional Test 6

Functional Test 7	
ID	F3.3
Description	See if tag statistics is updated properly when a new post is uploaded
Date	20.10.16.
Tester	Vigleik
Preconditions	App must be running, and user must be logged in. No other users is currently posting
Tasks	1. Make a note of the tag stats 2. Press scrapbook button 3. Press new Post button 4. Add text 5. Add at least one tag 6. Press Post button 7. Press back button 8. Compare the current stats with the numbers from task 1.
Result	The numbers next to the tag(s) from task 5 is now incremented by 1.

Table D.7.: Functional Test 7

D.1.4. Functional Testing, sprint 4

Functional Test 8	
ID	FT4.1
Description	Tries to generate student pairs
Date	20.10.16.
Tester	Ivar
Preconditions	App must be running, and user must be logged in as a teacher. The class must contain at least two students
Tasks	1. Press GetToKnow button 2. Press scrapbook button 3. Press generate new classification 3. Press choose exercise 4. Toggle an arbitrary exercise 5. Press save
Result	When the save button is pressed the user is redirected back to the teacher pairing view showing the updated grouping

Table D.8.: Functional Test 8

Functional Test 9	
ID	FT4.2
Description	Tries to report a post
Date	21.10.16.
Tester	Ivar
Preconditions	App must be running and user must have access to both a teacher account and a student account from the same class
Tasks	1. Log-in with a student account 2. Press scrapbook button 3. Press report button on an arbitrary post 3. Press side menu button 4. Press log-out 5. Log-in with a teacher account 6. Press scrapbook button 7. Find the post from task 3.
Result	The post is now highlighted red

Table D.9.: Functional Test 9

Functional Test 10	
ID	FT4.4
Description	Tries to hide a post
Date	21.10.16.
Tester	Ivar
Preconditions	App must be running and user must have access to both a teacher account and a student account from the same class
Tasks	1. Login with a teacher account 2. Press scrapbook button 3. Press hide button on an arbitrary post 3. Press sidemenu button 4. Press log-out 5. Log-in with a student account 6. Press scrapbook button 7. Find the post from task 3.
Result	The post is now hidden

Table D.10.: Functional Test 10

D.2. Paper Prototype

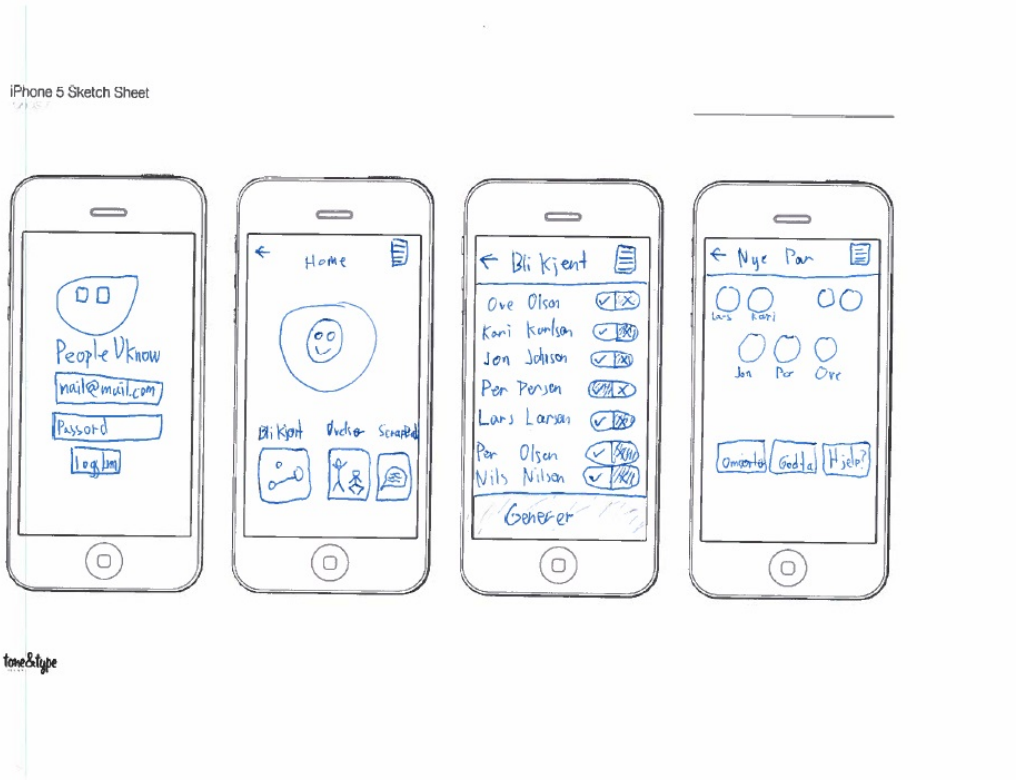


Figure D.1.: PeopleUKnow Application

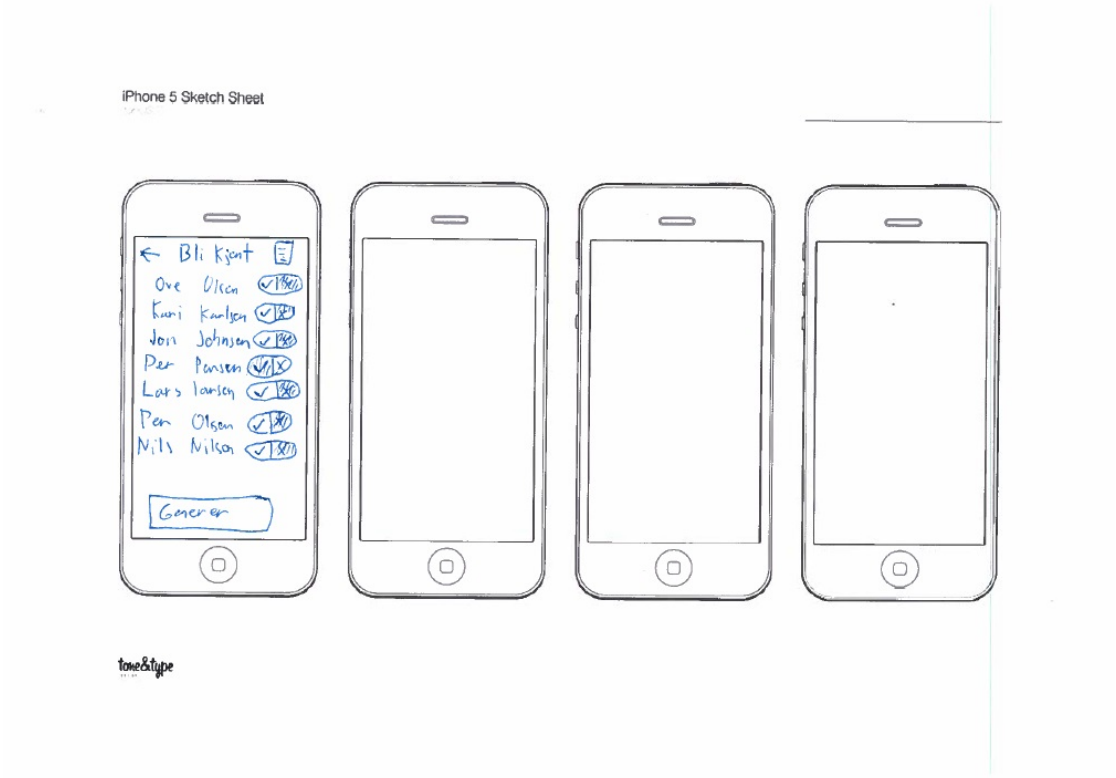


Figure D.2.: Get-to-know

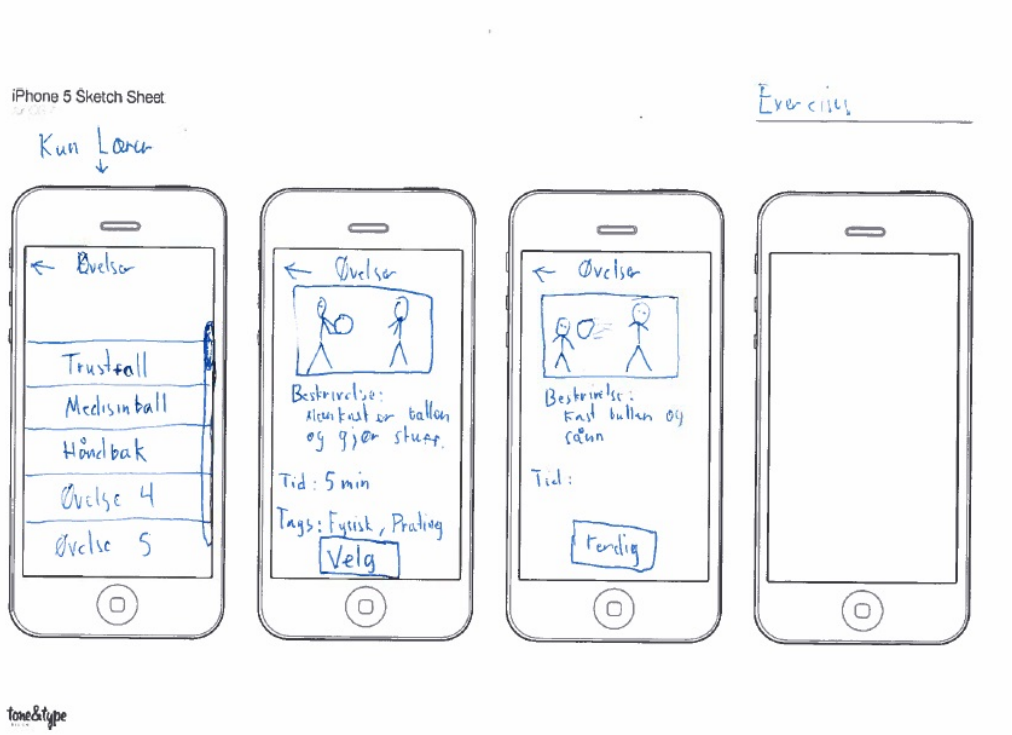


Figure D.3.: Exercise

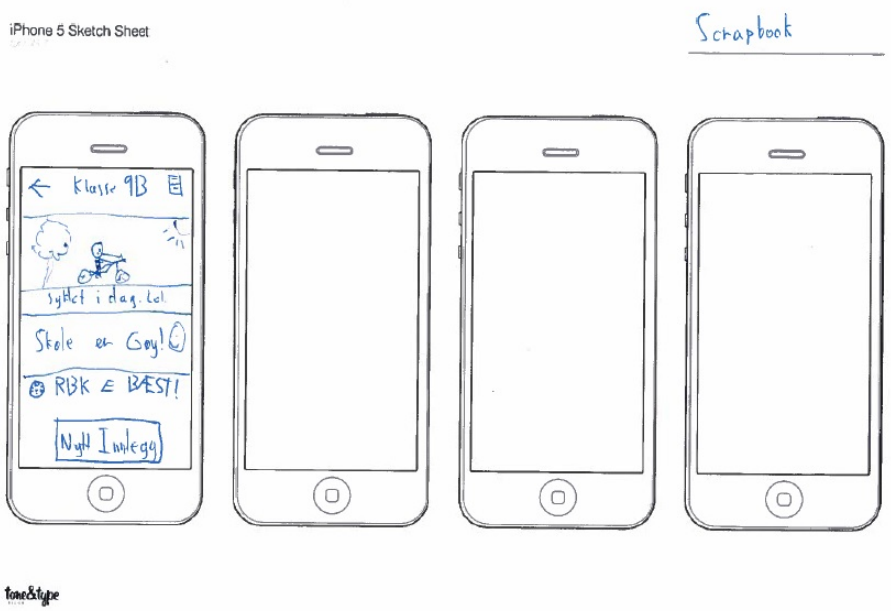


Figure D.4.: Scrapbook

D.3. Testplan for Usability

D.3.1. Formative

TestID	1
System	PeopleUKnow App
Description	Log in with an email and a password.
Requirement	
Stakeholder	Students, Teachers and Parents
Use Case	1
Input	Email and password
Success Criteria	Manage to log in
Failure Criteria	Not able to log in

Table D.11.: Usability Testplan 1

TestID	2
System	PeopleUKnow App
Description	View the exercise feature
Requirement	
Stakeholder	Students and Teacher
Use Case	7, 8
Input	
Success Criteria	User is able to navigate to Exercise
Failure Criteria	Not able to find exercise feature

Table D.12.: Usability Testplan 2

TestID	3
System PeopleUKnow App	
Description	After navigating to one of the features, then we want to navigate back to the overview of the application feature.
Requirement	
Stakeholder	Student and teacher
Use Case	12
Input	
Success Criteria	The user manage to get an overview of the application
Failure Criteria	Not able to find the overview

Table D.13.: Usability Testplan 3

TestID	4
System	PeopleUKnow App
Description	After getting assigned to a partner, it should be possible to view the partner.
Requirement	
Stakeholder	Student
Use Case	5
Input	
Success Criteria	The user is able to see his/her partner.
Failure Criteria	Not able to view the partner.

Table D.14.: Usability Testplan 4

TestID	5
System	PeopleUKnow App
Description	After someone has posted something on the scrapbook, it should be possible to view the scrapbook.
Requirement	
Stakeholder	Student
Use Case	2
Input	
Success Criteria	Able to view the scrapbook with post
Failure Criteria	Not able to view the scrapbook

Table D.15.: Usability Testplan 5

TestID	6
System	PeopleUKnow App
Description	In the scrapbook, it should be possible to comment on the posts
Requirement	
Stakeholder	Student and parents
Use Case	3
Input	Comment
Success Criteria	The comment has been posted on the post
Failure Criteria	Not able to comment on a post

Table D.16.: Usability Testplan 6

TestID	7
System	PeopleUKnow App
Description	Sharing on the scrapbook
Requirement	
Stakeholder	Student
Use Case	4
Input	
Success Criteria	Manage to share on the scrapbook
Failure Criteria	Not able to share on the scrapbook

Table D.17.: Usability Testplan 7

TestID	8
System	PeopleUKnow App
Description	When someone has posted a post on specific theme, it should be possible to search on that specific theme.
Requirement	
Stakeholder	Student and teacher
Use Case	9
Input	
Success Criteria	Able to search on post-related data
Failure Criteria	Not able to search on post-related data

Table D.18.: Usability Testplan 8

D.3.2. Summative

TestID	T1.1
Test Name	Notification on Not Successfully logged in
Test Type	Usability testing
Description	When user is not successfully logged in, the application notified the user
Goal	Get a notification when not manage to login
Requirement	F2
Stakeholder	Student, teacher
Input	Wrong user credentials (email or/and password)
Completion Criteria	User gets notification for not managing to login
Conditions	
Dependency	

Table D.19.: Test plan T1.1

TestID	T1.2
Test Name	Login
Test Type	Usability testing
Description	Test to see if a user can login with email and password
Goal	Be able to login
Requirement	F1
Stakeholder	Student, teacher
Input	Email and password
Completion Criteria	The user is logged in
Conditions	The user has already an account
Dependency	

Table D.20.: Test plan T1.2

TestID	T2
Test Name	View Post
Test Type	Usability testing
Description	Test to see if the user can see both pictures and text from other users from their class, which have been posted on the scrapbook, and see the styling chosen by classmate on the posts.
Goal	View post and styling from other classmate
Requirement	F3
Stakeholder	Student
Input	
Completion Criteria	
Conditions	User is logged in, the class has something on the scrapbook
Dependency	T1.2

Table D.21.: Test plan T2

TestID	T3
Test Name	Post Comment
Test Type	Usability testing
Description	Test to see if the user can comment on a classmate's post
Goal	Comment on a classmate's post
Requirement	F4
Stakeholder	Student
Input	Comment/text
Completion Criteria	The user manage to comment on the post, which is displayed
Conditions	User is logged in, the class has something in their scrapbook
Dependency	T1.2

Table D.22.: Test plan T3

TestID	T4
Test Name	Post and Style
Test Type	Usability testing
Description	Testing to see if a user can post pictures and text to the class scrapbook, and then choose the styling for the post
Goal	Post picture and text and choose the styling
Requirement	F5
Stakeholder	Student
Input	Picture, text and styling
Completion Criteria	The post with picture and text is posted with customized styling
Conditions	User is logged in
Dependency	T1.2

Table D.23.: Test plan T4

TestID	T5
Test Name	View Pairs
Test Type	Usability testing
Description	Test to see if the user can view which pair the user is assigned with by the teacher
Goal	See the assigned pair
Requirement	F6
Stakeholder	Student
Input	
Completion Criteria	The user manage to see his/her partner
Conditions	User is logged in, the teacher has assigned everyone a partner
Dependency	T1.2, T6.1, T6.2, T6.3

Table D.24.: Test plan T5

TestID	6.1
Test Name	Assign Pairs
Test Type	Usability testing
Description	Test to see if the teacher can create pairs
Goal	Create student pairs
Requirement	F7
Stakeholder	Teacher, student
Input	student names
Completion Criteria	The user manage to create student pairs
Conditions	User is logged in, there exist a class with students
Dependency	T1.2

Table D.25.: Test plan T6.1

TestID	6.2
Test Name	Modify Pairs
Test Type	Usability testing
Description	Test to see if the teacher can modify pairs
Goal	Modyfyinf the student pairs
Requirement	F7
Stakeholder	Teacher, student
Input	Student names
Completion Criteria	The user manage to modify the pairs as the user wats
Conditions	User is logged in, there exists a class with students, the pairs are already created
Dependency	T1.2, T6.1

Table D.26.: Test plan T6.2

TestID	T6.3
Test Name	Decide who is attending
Test Type	Usability testing
Description	Test to see if the teachers can decide which students are attending the class
Goal	Decide who is attending the class
Requirement	F7
Stakeholder	Teacher, student
Input	Student names
Completion Criteria	the user manage to decide which one is attending or not
Conditions	User is logged in, there exists a class with students, the pairs are already created
Dependency	T1.2 , T6.1

Table D.27.: Test plan T6.3

TestID	T7.1
Test Name	View all Exercises
Test Type	Usability testing
Description	Test to see if it is possible to view all the exercises
Goal	View all the exercises
Requirement	F8
Stakeholder	Teacher
Input	
Completion Criteria	Manage to view all the exercises
Conditions	User is logged in
Dependency	T1.2

Table D.28.: Test plan T7.1

TestID	T7.2
Test Name	Choose Exercise
Test Type	Usability testing
Description	Test to see if it is possible to choose an exercise that students in pairs are going to do
Goal	Choose exercise for student pairs
Requirement	F8
Stakeholder	Teacher, student
Input	Exercise
Completion Criteria	User manage to choose exercise for the student pairs
Conditions	User is logged in, student have pairs
Dependency	T1.2, T7.1

Table D.29.: Test plan T7.2

TestID	T8
Test Name	View Exercise
Test Type	Usability testing
Description	Test to see if the user can view the exercise, which is chosen by the teacher
Goal	View the exercise
Requirement	F9
Stakeholder	Student
Input	
Completion Criteria	The user manage to see the chosen exercise
Conditions	User is logged in, the students have a partner, the teacher has chosen the exercise
Dependency	T1.2, T6.1, T7.2

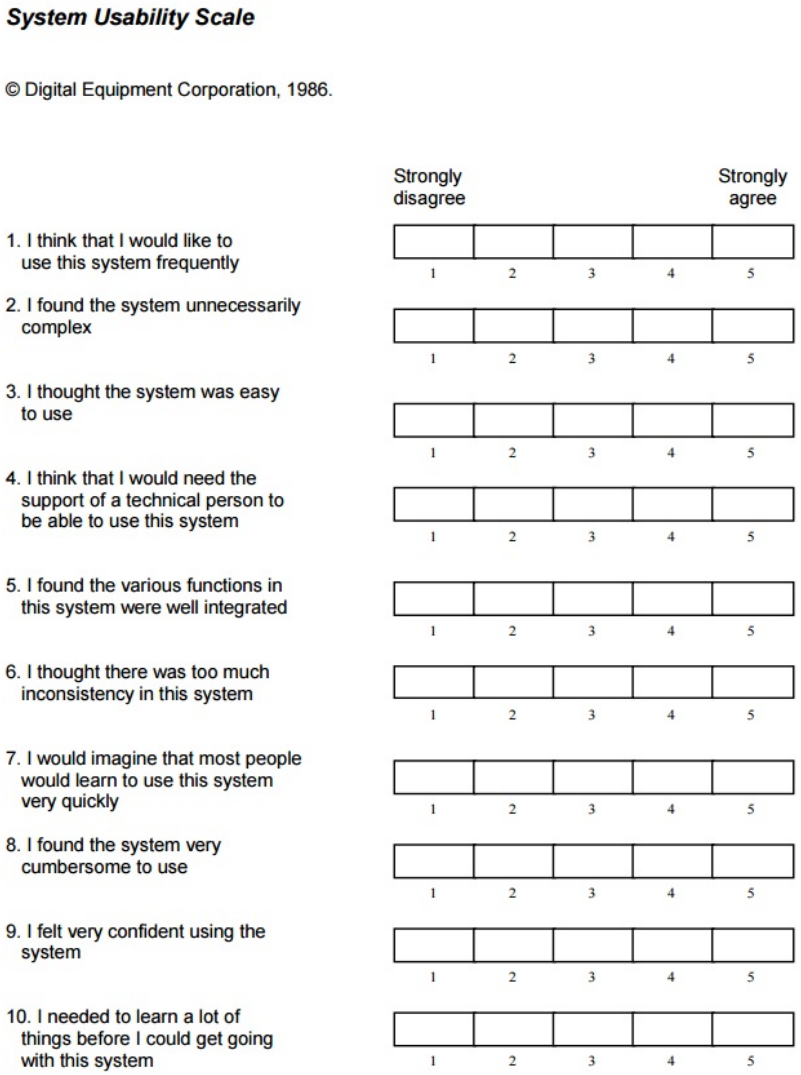
Table D.30.: Test plan T8

TestID	T9
Test Name	Profile Picture
Test Type	Usability testing
Description	Test to see if a user can edit profile picture
Goal	Change profile picture
Requirement	
Stakeholder	Student, teacher
Input	image
Completion Criteria	User has successful change profile picture
Conditions	User has a picture to change
Dependency	T1.2

Table D.31.: Test plan T9

TestID	T10
Test Name	Log out
Test Type	Usability testing
Description	Test to see if a user is able to logout from the application
Goal	Able to log out
Requirement	
Stakeholder	Student, teacher
Input	
Completion Criteria	User is log out from the application
Conditions	User is logged in
Dependency	T1.2

Table D.32.: Test plan T10



System Usability Scale

© Digital Equipment Corporation, 1986.

	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently	1	2	3	4	5
2. I found the system unnecessarily complex	1	2	3	4	5
3. I thought the system was easy to use	1	2	3	4	5
4. I think that I would need the support of a technical person to be able to use this system	1	2	3	4	5
5. I found the various functions in this system were well integrated	1	2	3	4	5
6. I thought there was too much inconsistency in this system	1	2	3	4	5
7. I would imagine that most people would learn to use this system very quickly	1	2	3	4	5
8. I found the system very cumbersome to use	1	2	3	4	5
9. I felt very confident using the system	1	2	3	4	5
10. I needed to learn a lot of things before I could get going with this system	1	2	3	4	5

Figure D.5.: SUS

D.3.3. Usability Test Result

Formative

We was able to find 3 different users to test the system. All of the user is studying in upper secondary high school, which is the user PeopleUKnow is trying to reach.

TestID	When	Tester	Expected Result	Comments
1	25.09	Thuy	Manage to login	Its look like a regular login page therefore it was easy to use.
2	25.09	Thuy	View Exercise	Easy to find. Could maybe include progressbar or something?
3	25.09	Thuy	Get an overview of application	Ok, pushed back button to get back to overview of the application.
4	25.09	Thuy	View Pairs	The user pushed the scrapbook for viewing pairs. The reason because it looked like a chat box where user can talk to each other. The user manage to view the pairs in the end. Strange that view pairs is in the get-to-know.
5	25.09	Thuy	View Scrapbook	Manage to find it.
6	25.09	Thuy	Comment on a post	The user pushed the new post function on the scrapbook page
7	25.09	Thuy	Sharing on the scrapbook	
8	25.09	Thuy	Search on Post-related data	User pushed the menu bar on the right corner.

Table D.33.: User Testing 1

TestID	When	Tester	Expected Result	Comments
1	25.09	Minh	Manage to login	Ok
2	25.09	Minh	View Exercise	Ok, asked what happens when pushing the profile picture
3	25.09	Minh	Get an overview of application	Ok
4	25.09	Minh	View Pairs	Ok
5	25.09	Minh	View Scrapbook	Ok
6	25.09	Minh	Comment on a post	How do I comment did the user asks? It does not say anything about it. User push the new post. And then the user push the picture
7	25.09	Minh	Sharing on the scrapbook	Ok
8	25.09	Minh	Search on Post-related data	Pushing the menu bar on the right

Table D.34.: User Testing 2

TestID	When	Tester	Expected Result	Comments
1	25.09	Quit	Manage to login	This user pushed the Iphone button. But after managed to log in.
2	25.09	Quit	View Exercise	Push the profile picture. But then manage to push the right button.
3	25.09	Quit	Get an overview of application	Push the back button, since it is not any other buttons to push.
4	25.09	Quit	View Pairs	Ok
5	25.09	Quit	View Scrapbook	Ok
6	25.09	Quit	Comment on a post	Pushing the picture.
7	25.09	Quit	Sharing on the scrapbook	Pushing the button on the right corner. The user don't know where things are.
8	25.09	Quit	Search on Post-related data	Pushing the picture.

Table D.35.: User Testing 3

Summative

TestID	When	Test Obj.	Expected Result	Comments
T1.1	26.10	student1	Gets error notification	When user tried to enter in password, user tried to swipe down to see the password field. After some tries user manage to click on the keyboard return and finally manage to type in password.
T1.2	26.10	student1	Manage to login	ok
T2	26.10	student1	View post	User clicked first on exercise, after user tries to click on different tags and later manage to click on scrapbook.
T3	26.10	student1	Comment on post	ok
T4	26.10	student1	Create a new post with styling	Created a post without picture.
T5	26.10	student1	View pair	ok
T8	26.10	student1	View exercise	User did not know where to find the exercise so user clicked on "drømmer" and then "aktiviteter". User then tried menu bar and then manage to find week buddy in getToKnow.
T9	26.10	student1	Take a new profile picture	User manage to take a profile picture, but forgot to save the picture in the end.
T10	26.10	student1	Log out	Found the button since the user had navigated to menu bar before.

Table D.36.: User Testing 1

Feedback from student 1

- Have some description on the menu button. So far the button is only . . . (dash dash dash).
- It should be possible when clicking on the picture
- Would use the application, but user said that the user needed some help using it before user could use it by him self.

TestID	When	Test Obj.	Expected Result	Comments
T1.1	26.10	student2	Get error notification	Tried to scroll down to type in password.
T1.2	26.10	student1	Manage to login	ok
T2	26.10	student2	View post	User tried to click on every tags, exercise and then found Scrapbook.
T3	26.10	student2	Comment on post	ok
T4	26.10	student2	Create a new post with styling	Wondering if it was possible to see picture before posting.
T5	26.10	student2	View pair	Found this because user had manage to click on this earlier.
T8	26.10	student2	View exercise	ok
T9	26.10	student2	Take a new profile picture	User clicked on the profile picture, but then manage to navigate to side menu and then changed profile picture.
T10	26.10	student2	Log out	ok

Table D.37.: User Testing 2

Feedback from student 2

- Should be able to see which tags that user has choosen.
- Menu button should be more down
- Maybe the post should be sorted by tags
- It was easy to use
- Liked the side menu part
- Maybe should change the name of the side menu button
- Wanted it should be possible to change the color of the profile picture circle

TestID	When	Test Obj.	Expected Result	Comments
T1.1	26.10	student3	Gets error notification	Tried to scroll up to manage to see password field
T1.2	26.10	student3	Manage to login	ok
T2	26.10	student3	View post	Tried to click on "drømmer", "aktiviteter", and profile picture. Found the Scrapbook in the end. At this step something went wrong with Firebase. But we continued the test without the pictures
T3	26.10	student3	Comment on post	ok
T4	26.10	student3	Create a new post with styling	User asked where the picture should come(At this stage it was something wrong with firebase)
T5	26.10	student3	View pair	User tried to click on tags such as "samarbeid", then user tried to click on side menu, in the end user gets help wth findig exercise
T8	26.10	student3	View exercise	ok
T9	26.10	student3	Take a new profile picture	User tries menu bar button, then user tries to push on profile picture. And when user tried to save the picture the applicatio stopped and work. The profile picture did not get saved.
T10	26.10	student3	Log out	ok

Table D.38.: User Testing 3

Feedback from student 3

- The application was really good

TestID	When	Test Obj.	Expected Result	Comments
T1.1	26.10	student4	Gets error notification	
T1.2	26.10	student4	Manage to login	
T2	26.10	student4	View post	
T3	26.10	student4	Comment on post	
T4	26.10	student4	Create a new post with styling	
T5	26.10	student4	View pair	
T8	26.10	student4	View exercise	
T9	26.10	student4	Take a new profile picture	
T10	26.10	student4	Log out	

Table D.39.: User Testing 4

Feedback from student 4

TestID	When	Test Obj.	Expected Result	Comments
T1.1	26.10	student5	Gets error notification	User tries to scroll up to see password field.
T1.2	26.10	student5	Manage to login	ok
T2	26.10	student5	View post	user use 4-6 secongs before user click on Scrapbook
T3	26.10	student5	Comment on post	ok
T4	26.10	student5	Create a new post with styling	User tries to swipe up and then click on "nytt innlegg". User manage to click on tags button two times after already adding a tags.
T5	26.10	student5	View pair	Tries to push on "samabeid" and since the other tags is not buttons, the user wont push anything other. After user push on exercise button.
T8	26.10	student5	View exercise	ok
T9	26.10	student5	Take a new profile picture	User clikcs on the circle, exercise and then clicks on the profile picture in the exercise, user goes to the menu bar and tries to puch on the profile cirle of week buddy. In the end user manage to see "bytt profilbilde" in the menu bar.
T10	26.10	student5	Log out	User clicks on menu buttons, since user found it from last step.

Table D.40.: User Testing 5

Feedback from student 5

- Conflict between English and Norwegian
- Menu button should be placed on a place where it is more natural or it should be more clearer since it overlapp the battery button. The menu button looks like a facebook messenger buton.
- Maybe more functionality

TestID	When	Test Obj.	Expected Result	Comments
T1.1	26.10	student6	Gets error notification	Tried to scroll down to see the password field.
T1.2	26.10	student6	Manage to login	ok
T2	26.10	student6	View post	Wonder if it was scrapbook and manage to find post.
T3	26.10	student6	Comment on post	
T4	26.10	student6	Create a new post with styling	
T5	26.10	student6	View pair	
T8	26.10	student6	View exercise	
T9	26.10	student6	Take a new profile picture	
T10	26.10	student6	Log out	

Table D.41.: User Testing 6

Feedback from student 6

D.3.4. SUS result

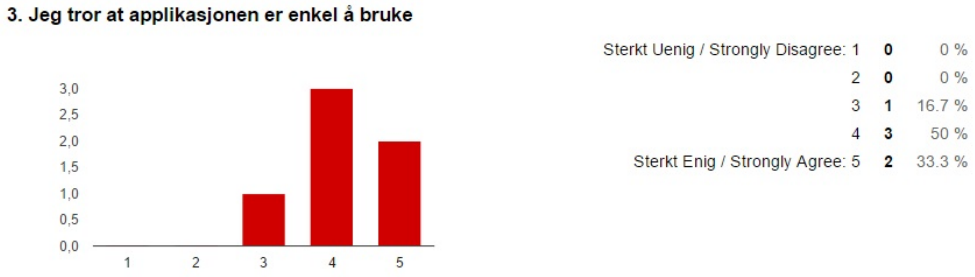


Figure D.6.: I would imagine that most people would learn to use this system very quickly

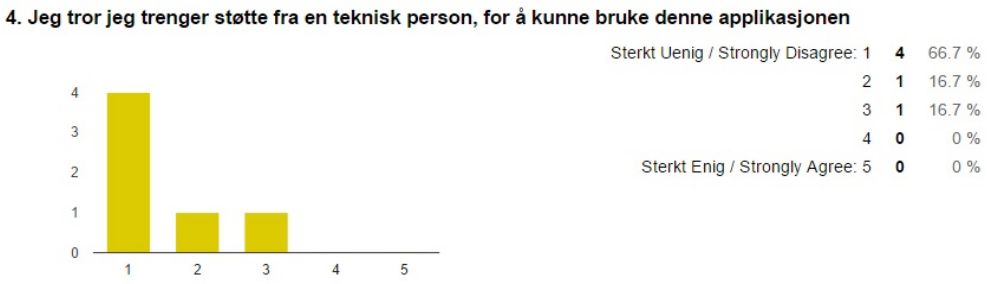


Figure D.7.: I think that I would need the support of a technical person to be able to use this system

Figure D.8.: I found the various functions in this system were well integrated

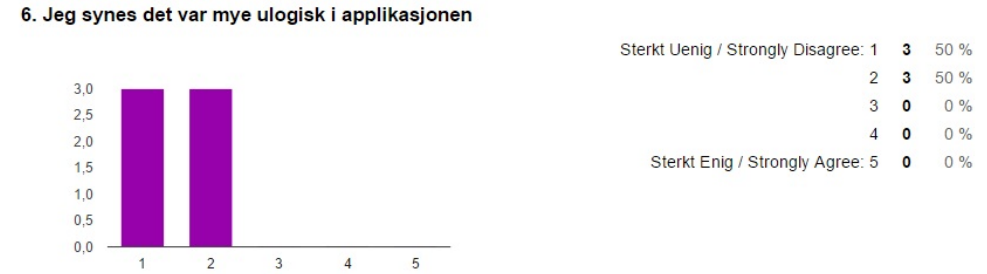


Figure D.9.: I thought there was too much inconsistency in this system

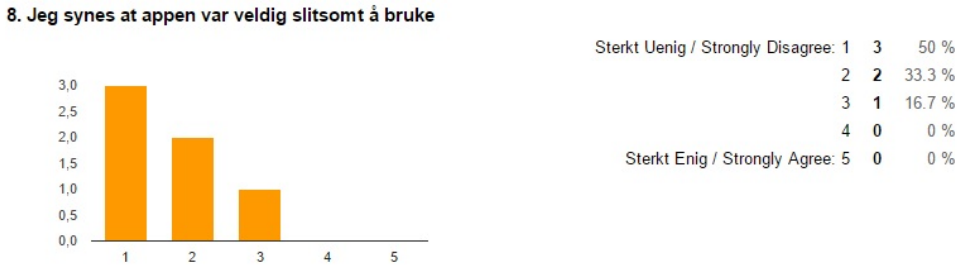


Figure D.10.: I found the system very cumbersome to use

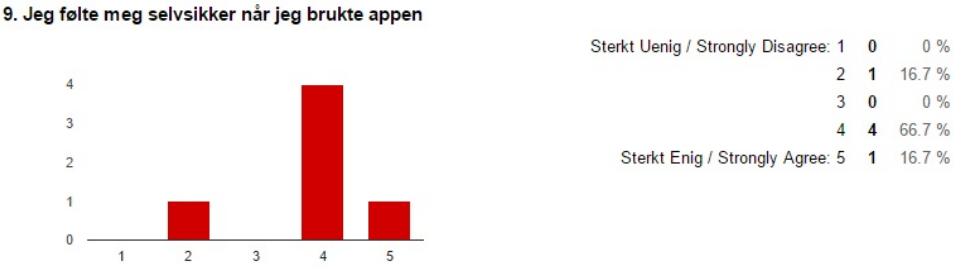


Figure D.11.: I felt very confident using the system

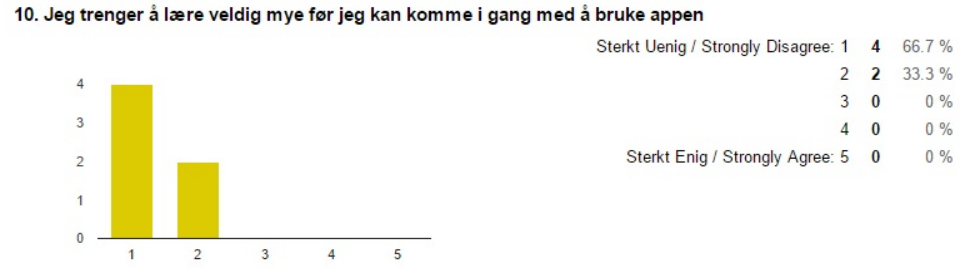


Figure D.12.: I needed to learn a lot of things before I could get going with this system

Bibliography

[1] Ambient. After school - funny anonymous school news for confessions and compliments. https://itunes.apple.com/us/app/after-school-funny-anonymous/id918396645?mt=8, 2016.

[2] Apache. Cordova. https://cordova.apache.org/, 2016.

[3] Atom. Atom ide. https://atom.io/, 2016.

[4] Miller Bangor, Kortum. Sus research paper. http://uxpajournal.org/wp-content/uploads/pdf/JUS_Bangor_May2009.pdf, 2009.

[5] bullyingstatistics.org. Bullying statistics. http://www.bullyingstatistics.org/content/bullying-and-suicide.html, 2016.

[6] Difi. Risikoanalyse - mal. https://www.anskaffelser.no/verktoy/risikoanalyse-mal, 2015.

[7] draw.io. draw.io. https://www.draw.io/, 2016.

[8] ESDoc. Esdoc. https://esdoc.org/, 2016.

[9] Facebook. nuclide. https://nuclide.io/, 2016.

[10] Facebook. React native. https://facebook.github.io/react-native/, 2016.

[11] Facebook. React native. https://facebook.github.io/react-native/, 2016.

[12] Facebook.com. Facebook. https://www.facebook.com/, 2016.

[13] Feide. Feide. https://www.feide.no/virkemate, 2016.

[14] Firebase. Integrate with google cloud platform. https://firebase.google.com/docs/storage/gcp-integration, 2016.

[15] Firebase. Integrate with google cloud platform. https://firebase.google.com/docs/database/security, 2016.

[16] Github.com. Github. https://github.com/, 2016.

[17] Google. Firebase. https://firebase.google.com/, 2016.

[18] Google. Google drive. https://drive.google.com/, 2016.

[19] https://xtensio.com/user persona/. User personas. https://xtensio.com/user-persona/, 2016.

[20] NTNU IDI. Course compendium appendix. http://www.idi.ntnu.no/emner/tdt4290/docs/TDT4290-compendium-2016.pdf, 2016.

[21] Henrik Kniberg. Scrum. https://www.crisp.se/file-uploads/Kanban-vs-Scrum.pdf, 2016.

[22] Philippe Kruchten. 4+1 view model. http://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf, 1995.

[23] Philippe Kruchten. 4+1 view model. http://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf/, 1995.

[24] Deborah Lessne and Incorporated (SEI) Melissa Cidade of Synergy Enterprises. Student reports of bullying and cyber-bullying: Result from the 2013 schoold crime supplement to the national crime victimization survey. http://nces.ed.gov/pubs2015/2015056.pdf, 2016.

[25] Lovdata.no. Lov om behandling av personopplysninger. https://lovdata.no/dokument/NL/lov/2000-04-14-31#KAPITTEL_2, 2016.

[26] Marvel. Marvel. https://marvelapp.com/, 2016.

[27] Gary McGraw. The 7 touchpoints of secure software. http://www.drdobbs.com/the-7-touchpoints-of-secure-software/184415391, 2016.

[28] Microsoft. The stride threat model. https://msdn.microsoft.com/en-us/library/ee823878%28v=cs.20%29.aspx, 2016.

[29] Lorraine Millan. The social navigator. http://www.socialnavigatorapp.com/social_navigator.php, 2016.

[30] nobullying.com. What are the causes of bullying. https://nobullying.com/what-are-the-causes-of-bullying/, 2016.

[31] Rhine o Enterprises LLC. Team shake. https://itunes.apple.com/us/app/team-shake/id390812953?mt=8, 2016.

[32] owasp. op 10 2014-i2 insufficient authentication/authorization. https://www.owasp.org/index.php/Top_10_2014-I2_Insufficient_Authentication/Authorization, 2016.

[33] Karen N. Peart. Bullying-suicide link explored in new study by researchers at yale. http://news.yale.edu/2008/07/16/bullying-suicide-link-explored-new-study-researchers-yale, 2016.

[34] PeopleUKnow. Peopleuknow websitel. http://peopleuknow.no, 2016.

[35] Margaret Rouse. Use case. http://searchsoftwarequality.techtarget.com/definition/use-case, 2016.

[36] Jeff sauro. Sus research. https://www.measuringu.com/article.php?uname=sus.php, 2011.

[37] ShareLatex.com. Sharelatex. https://www.sharelatex.com/, 2016.

[38] Dag Sjøberg. Kravhåndtering. http://www.uio.no/studier/emner/matnat/ifi/INF1050/v14/timeplan/inf1050.krav.29.1.2014.pdf, 2016.

[39] Slack.com. Slack. https://slack.com/, 2016.

[40] SmashingMagazine. Smashingmagazine. https://www.smashingmagazine.com/wp-content/uploads/2016/06/02-react-native-architecture-opt.png, 2016.

[41] socialskillbuilder. Social skill builder: My school day. https://itunes.apple.com/us/app/social-skill-builder-my-school/id570787918?mt=8, 2016.

[42] Everyday Speech. Let's be social: Social skills development. https://itunes.apple.com/us/app/lets-be-social-social-skills/id772244049?mt=8, 2016.

[43] Sublime. Sublime. https://www.sublimetext.com/, 2016.

[44] Trello.com. Trello. https://trello.com/, 2016.

[45] Inger Anne Tøndal. Protection poker: Spill deg til bedre programvaresikkerhet! http://infosec.sintef.no/informasjonssikkerhet/2016/05/protection-poker-spill-deg-til-bedre-programvaresikkerhet/, 2016.

[46] Wikipedia. Model view controller. https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller, 2016.

[47] Wikipedia. Waterfall method. https://en.wikipedia.org/wiki/Waterfall_model, 2016.

[48] www.crisp.se. Planning poker. https://www.crisp.se/bocker-och-produkter/planning-poker, 2016.

[49] Xamarin. Xamarin. https://www.xamarin.com/, 2016.

[50] Youtube. Statistics. https://www.youtube.com/yt/press/statistics.html, 2016.