

## Modernizing an Application

The process of replicating and modernizing Norske Sanitetskvinner's web application

### GRUPPETID:

Andersen, Henriette  
Risdalen, Bjørnar  
Sjursen, Hanne P.  
Staurheim, Trym E.

### SUPERVISOR

Nilsen, Hallgeir

**University of Agder, 2021**

Faculty of Social Sciences

Department of Information Systems

## **Preface**

This report is a presentation of the work we have done in our last semester at the bachelor's program at IT and information system and a representation of the competence and knowledge we have acquired over the past three years at the University of Agder. In that regard, we want to thank the people that have helped us along the way to accomplish a successful project.

First off, we want to thank Knowit Sør for giving us the opportunity to work for and cooperate with them on this project. Knowit has offered their help and advice throughout the project, especially in the initial phase when we needed it the most. They have also been a positive resource for us when we have asked for help on other tasks and assignments.

We would like to give our supervisor and Product Owner Thomas Andréé Wang a special thanks for giving us his guidance and valuable tips throughout the project. He has been both flexible and helpful when we have been stuck in pipelines and design choices and given us his best recommendations and practices to help us improve both the process and the product. We are highly appreciative of his effort to make us succeed in the project.

We would also like to thank our supervisor at UiA, Hallgeir Nilsen, for always being available for supervision, staying favorable to the team's effort, and providing us with valuable feedback during both the project and throughout the last three years at the university.

At last, we want to thank our families, friends, and better halves for supporting us, being available for us when we needed to cry, laugh, or rant - both over the project and life in general. This has been of great importance to help us keep our motivation running when times have been challenging and critical for us to succeed in the project.

## *Table Of Content*

<b>1.0 Introduction</b>	<b>6</b>
1.1 Overview	6
1.1.1 About Knowit Sør	7
1.1.2 User group: Norske Kvinners Sanitetsforening	7
1.2 Purpose of project	8
1.2.1 Characteristics of the Project	8
<b>2.0 Central decisions</b>	<b>8</b>
2.1 Requirements from Knowit	9
2.1.1 Project management requirements	9
2.1.3 Technological requirements	9
2.1.4 Quality Requirements	10
2.2 Quality Management	10
2.2.1 Internal and external quality	10
2.2.2 Quality Assurance	11
2.2.3 Technological decisions	14
2.2.4 Design principles	15
2.3 Project Management Choices	16
2.3.1 Azure DevOps	16
2.3.2 Scrum	17
2.3.3 Risk Management	17
2.3.4 Backlog, Estimation & Priorities	17
2.3.5 Communication	19
2.3.6 Roles in the project	20
2.4 Other central decisions in the project	21
2.4.1 Development Role Distribution	21
<b>3.0 Running the project</b>	<b>21</b>
3.1 Planning and Analysis	22
3.1.1 Planning during Covid-19	22
3.1.2 Planning administrative tasks	22
3.1.3 Product Backlog	23
3.2 Project Management	24
3.2.1 Azure DevOps	24
3.2.2 Estimation of the backlog	24
3.2.3 Burndown Chart	26
3.2.4 Risk Management	26
3.2.5 Communication	27
3.2.6 Roles in the Project	28
3.3 Agile Methodology	28
3.3.1 Scrum Team	29
3.3.2 Sprint Planning	29
3.3.3 Sprint Backlog	29
3.3.4 Daily Standup	30
3.3.5 Sprint Review	31
3.3.6 Sprint Retrospective	32
3.4 Design	32

3.4.1 Wireframes	33
3.4.2 WCAG	33
3.4.3 Benyons Design Principles	34
3.5 Development Process	39
3.5.1 The Use of Technology	39
3.5.2 Code Review	41
3.5.3 Design Patterns	42
3.5.4 Version Control	44
3.5.5 Code Testing	44
3.5.6 CI/CD	45
3.5.7 Pair Programming	46
<b>4.0 The product</b>	<b>47</b>
<b>5.0 Reflection</b>	<b>47</b>
5.1 Project Management	47
5.1.1 Agile Methodology	47
5.1.2 Risk management	48
5.1.3 Estimation and Prioritization	48
5.1.4 Communication	49
5.2 Quality of End Product	50
5.3 Challenges	51
5.3.1 Covid-19	51
5.3.2 Roles and Role Distribution	52
5.4 Summarization	53
<b>6.0 Statement from Client</b>	<b>54</b>
<b>7.0 Self evaluation</b>	<b>55</b>
<b>References</b>	<b>57</b>
Appendix 1: Group contract	63
Appendix 2: Wireframes	66
Appendix 3: Risk matrix	69
Appendix 4: Sprint review	71
Appendix 5: Sprint retrospective	77
Appendix 6: Code standards	80
Appendix 7: Git Procedures	82
Appendix 8: Structuring of component interfaces.	84
Appendix 9: Tailwind config file and example	85
Appendix 10 - Azure DevOps	86
Appendix 11 - Test examples	89

*Figure list*

<b>Figure 1: Application flow diagram .....</b>	<b>5</b>
<b>Figure 2: Existing technology .....</b>	<b>6</b>
<b>Figure 3: Git Flow .....</b>	<b>12</b>
<b>Figure 4: Example of Epic-Feature-User Story-Task .....</b>	<b>17</b>
<b>Figure 5: Discord window with different channels on the left, and content of the selected channel on the right .....</b>	<b>19</b>
<b>Figure 6: User stories .....</b>	<b>22</b>
<b>Figure 7: Estimation of Lokalforening component .....</b>	<b>24</b>
<b>Figure 8: Burndown Chart .....</b>	<b>25</b>
<b>Figure 9: Risk Matrix .....</b>	<b>26</b>
<b>Figure 10: Sprint backlog .....</b>	<b>29</b>
<b>Figure 11: Sprint review form used for preparation (Mendez, 2015) .....</b>	<b>31</b>
<b>Figure 12: Sprint retrospective table from 01.03.2021 .....</b>	<b>31</b>
<b>Figure 13: Wireframe of login page .....</b>	<b>32</b>
<b>Figure 14: Example of visibility - existing version vs. our version .....</b>	<b>34</b>
<b>Figure 15: Example of familiarity .....</b>	<b>34</b>
<b>Figure 16: Example of “sticky” footer .....</b>	<b>35</b>
<b>Figure 17: Example of the use of constraints .....</b>	<b>36</b>
<b>Figure 18: TypeScript type inference and type declaration .....</b>	<b>38</b>
<b>Figure 19: A model of the whole application .....</b>	<b>41</b>
<b>Figure 20: Pipeline yaml in browser editor .....</b>	<b>44</b>

## 1.0 Introduction

In the bachelor course IS-304, the assignment was to carry out an IT/IS-related project using established methods and techniques, including planning, estimating, performing testing, following up, and documenting the process. We were also supposed to define quality in the project and implement measures that assure quality and control progress and quality throughout the project (UiA, 2020). Our assignment was to modernize the code stack for an already existing system for the voluntary organization: Norske Kvinners Sanitetskvinnene. To carry out this project we were employed by an IT consultant company: Knowit Sør.

In this chapter, we will present the overview and purpose of our project. In the following sections, we will introduce the system, the characteristics, requirements of our system, the user group for the system, and our product owner: Knowit Sør. In the rest of the report, we will differentiate between the terms *system* and *project*, whereof we define the word system as: “the delivery of the product” and project as: “the delivery of the product plus the processes surrounding project management.” The terms *system* and *application* will be used interchangeably.

## 1.1 Overview

Knowit Sør assigned us a task regarding rewriting the full stack codebase for an already existing system for Norske Kvinners Sanitetsforening. The existing system is an internal application for registering and updating information regarding local associations within Norske Kvinners Sanitetsforening, resulting in an annual report generated in Excel. The application assumes that the users have login information, and there are different access levels based on whether you are an administrator or a regular member. In addition to the login page, the application includes the views as following:

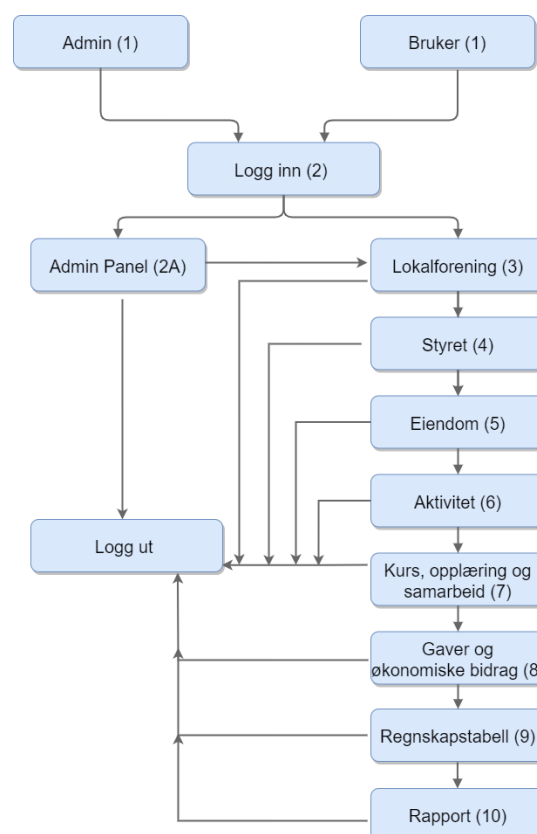
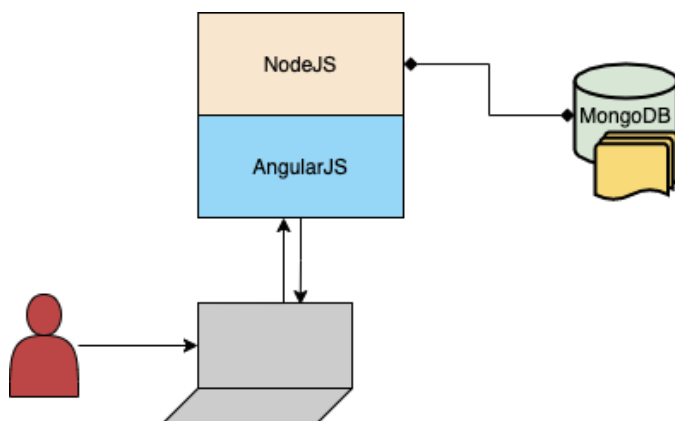


Figure 1: Application flow diagram

administrator panel, local association and member information, the board, real estate, activity, courses and training, gifts and financial contributions, accounting figures, and report. In the last view, all the registered information may be downloaded as a portable document format (PDF). The flow of the application is presented in figure 1.

The existing system was written in these technologies: Angular with JavaScript (frontend), Node.js (backend), and MongoDB (database). Our task was to replace these with React with TypeScript (frontend), Java (backend), and PostgreSQL (database) while also retaining the same functionality as in the existing application. To summarize: the task was to create a replica of the already existing system, with the possibility of being creative, especially regarding the design.



**Figure 2:** Existing system stack

### 1.1.1 About Knowit Sør

Knowit Sør is an IT consultant company in the consultant group Knowit AB. They provide and implement solution-type services with tailored expertise in system development and continuous integration- and delivery (CI/CD) to their customers (Knowit, 2020). Knowit Sør takes significant steps in understanding the business processes and problem domains of their customers to facilitate domain-specific solutions. Knowit is the product owner and central stakeholder of this project. We were assigned an employee, Thomas Andréé Wang, from Knowit that functioned as both our supervisor and product owner in our project.

### 1.1.2 User group: Norske Kvinners Sanitetsforening

The system's users are the members of “Norske Kvinners Sanitetsforening” (NKS). NKS is the most prominent female association in Norway, promoting women’s rights and healthcare

with over 40.000 members spread across 600 local associations (Norske Kvinnens Sanitetsforening, n.d). Our user group has these characteristics: Female, primarily elderly users (60+), and can be categorized as super users and regular users. Superusers are those already familiar with the old system and use it regularly. Regular users are those who use the application annually.

## **1.2 Purpose of the project**

The goal for our project was to conduct an IS-relevant project that fulfilled the learning objectives of IS-304, as well as contributing to learning, both for us as a team and individually. From the stakeholder's (Knowit) point of view, the goal was to change the existing stack into their in-house technology stack. Another aspect was to give us a project where we could apply our knowledge and experience from the past three years, which could potentially surpass the quality of the existing system and be cost-efficient for both Knowit and NKS.

Knowit also focused on facilitating the process for our team through version control, project management, and documentation using Azure DevOps. This project also acts as a way for us to acquire valuable experience in running an actual project, especially regarding the system development process.

### **1.2.1 Characteristics of the Project**

There are several characteristics of the project that made it rather unique. Firstly, the product was to be a replica of the old system entailing that we need to support the same actions, store the same data, and provide the same functionality. Secondly, we worked with an Agile workflow using MoSCoW prioritization, but because of the replica nature of the product, we found ourselves with more must-haves than usual as every feature had to be completed for the product to be used. Third and finally, the project is heavily influenced by the users of the product, who are already familiar with the current system and may be resistant to significant changes.



## **2.0 Central decisions**

In the following chapter, we will present the central decisions on methodology, quality, technological decisions, project management, and other central decisions that have been made throughout this project.

### **2.1 Requirements from Knowit**

Knowit had several requirements that impacted our central decisions on methodology, project management, technology, standards, quality, and quality assurance.

#### **2.1.1 Project management requirements**

For project management, Knowit required an agile methodology in conjunction with the project management suite Azure DevOps. It was also the agile methodology Knowit themselves was familiar with and preferred in their in-house system development.

Azure DevOps provides project management services that facilitate collaboration, building, deploying, and covering the entirety of the development cycle (Comley et al., 2021). At first, Knowit wanted us to test the features of Azure DevOps. Nevertheless, within a week of the preliminary meeting in January, Knowit had already decided to start using it.

The agile methodology in software development bases its fundamentals on the agile manifesto from 2001 and is centered around the idea of iterative development (AgileManifesto, n.d., Cprime, n,d). Although Knowit chose this methodology for us, we would have made the same decision ourselves, as there are several benefits to working agile, contrary to non-agile methodologies. For one, it allows for frequent adaptation and introspection into the development process, which increases control, and facilitates that the right priorities are being made. Secondly, it puts the customers in the development cycle, creating a loop based on feedback to keep the development focused towards the customers' goals. Third, agile methodology is the most used methodology according to a survey conducted by Stack Overflow (Stack Overflow, 2018), and according to PWC, agile projects are 28 percent more successful (Jonnalagadda et al., 2017, p. 1).

#### **2.1.3 Technological requirements**

Knowit's current in-house technological stack consists of Java with Spring, React with TypeScript, along with a PostgreSQL database; subsequently, we were required to use the

same stack. Java coupled with Spring - which is used by large companies such as Netflix (T. Wickse, 2019) - makes for a powerful tool suited for both large- and small-scale applications. React is a Component-Based JavaScript library for building user interfaces (React, n.d.) and is one of, if not the most, popular UI libraries - tagged in one in every twenty Stack Overflow posts (Stack Overflow, 2021). PostgreSQL is a powerful, open source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads (PostgreSQL, n.d.).

#### **2.1.4 Quality Requirements**

Regarding quality and assurance, Knowit wants a replica of the existing system and source code that is easier to maintain from a system maintenance standpoint. Moreover, they require that (1) the development process is good, (2) that the source code is thoroughly tested with at least 70 percent of the codebase covered to assure its quality, (3) and that we establish a build and deploy regime.

*What is a good development process?*

According to Feiler & Humphrey, development processes are contextually dependent on the project, and therefore it is difficult to define what makes a development process “good”, suggesting that each project must find the process that befits its needs (Feiler & Humphrey, 1993). This is in line with Benediktsson et al., who suggests that a good development process is contingent on the context (Benediktsson et al., 2006). Contextually, our project has precise requirements both regarding methodology and technical specifications. Thus, adhering to the agile methodology and implementing its practices, including; control through dialogue and planning with Knowit, confirming priorities, estimated time for delivery (ETD), and most importantly, being able to adapt to changing needs and specifications should result in the process being termed “good”.

## **2.2 Quality Management**

This chapter presents central decisions on quality and quality assurance and aspects that we consider impacting quality; Internal & External quality, version control, code standards, testing, and communication.

### **2.2.1 Internal and external quality**

Our project has two perspectives concerning quality: internal and external quality.

We define *Internal quality* as complying with Knowit's requirements, including automation, maintainability, and structure. With automation, we had to establish a build and deploy regime with automated tests. Maintainability involves writing code that is easy to maintain and has high test coverage, and structure entails a clean and organized source code repository.

We define *External quality* as meeting the requirements specified by Knowit's client and encompasses elements like correctness, completeness, and consistency. Correctness means that the calculations and data handled by the system are correct; Completeness is that the system adheres to the client's tailored needs; Consistency annotates that the design is uniform and consistent.

### **2.2.2 Quality Assurance**

We have implemented several measures to assure both the internal and external quality we stipulated throughout the project. Assuring internal and external quality has been verified through frequent dialogue with our product owner and has been continuously reviewed through Sprint retrospectives- and reviews. Moreover, measures have been taken to proactively assert quality in regards to the development life-cycle by implementing code standards, design patterns, reviews, testing, and version control.

#### *Code standards*

To ensure quality in the source code, we decided to create a coding standard for the group to follow. The standard covered naming conventions for classes, components, and variables, including spacing, code-indentation, and folder structure. In more detail, we chose to split the frontend and backend into two different repositories - resulting in the source code being uniform and comprehensible, and maintainable for the entire group. See Appendix 1 for the formal code standards.

#### *Design Patterns*

Another aspect in regards to facilitating quality is the implementation of design patterns in the source code. A design pattern provides a reusable solution for common design problems that typically occur in designing software (Barnes & Kölling, 2017, p. 542). We have followed industry standards for both front- and backend development. We have used the Singleton pattern and the Composite pattern for the frontend, which couples nicely with the React

library, resulting in reusable components that enable loose coupling, high cohesion, and localizing change.

For the backend, we have used a Domain-Driven Design (DDD) pattern combined with the Repository pattern (Albano, 2020) interfaced by Representational State Transfer (REST) over the HTTP protocol. Both are typical patterns that befit the structure of Spring Boot. The pattern is divided into three layers; (1) the controller layer, which receives HTTP requests and passes them to (2) the service layer responsible for the business logic of the application, which in turn passes domain objects or requests to (3) the repository layer which persists or retrieve data to the database.

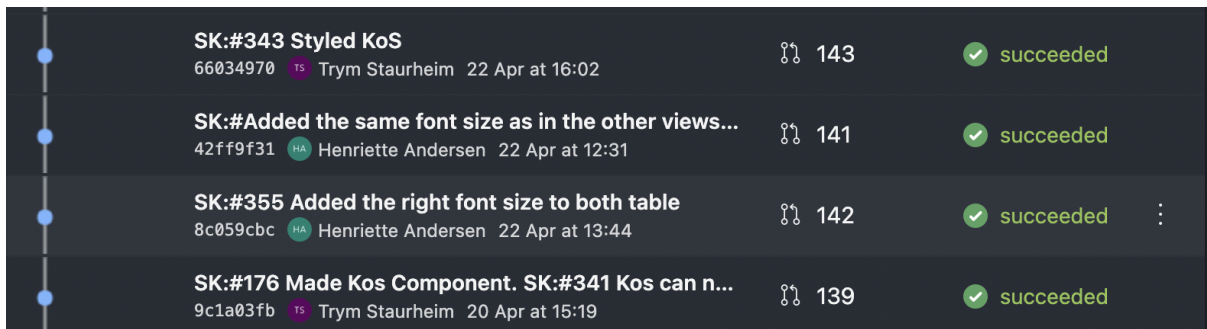
### *Code Review*

To further ensure quality in source code, we implemented a formal list to standardize reviewing source code, ensuring that every team member knew how to make a pull request, when to make a new branch, and how to merge and fix feedback from the reviewers of the source code. It was important for us as individual programmers, as well as Knowit, that we put effort into reviewing each other's source code. As a measure to this implication, we chose to have two required reviewers of each pull request besides the author. These decisions regarding code review were to ensure quality in the code, as this, in the end, would ensure quality in the whole application and that bugs were being fixed continuously. Both the general and specific procedures connected to the Git Flow are described in Appendix 2.

### *Version Control*

For version control, we used the integrated view in Azure DevOps that incorporates version control software through the use of Git. To facilitate control over the produced source code, we decided to follow the Git Flow WorkFlow (Atlassian, n.d.). Git Flow expedites for version control through strict structure of branches and merging of source code through pull requests. Primarily by ensuring that the `Main` branch has only production-ready source code and using Gits branching feature to maintain a stable and rollbackable source code. Each task in the backlog should be a separate branch in git, and when the task is completed, the branch is then rebased onto the current production-ready source code to keep a linear version history. Below

is an example of the linear history of our main branch.



SK:#343 Styled KoS 66034970 TS Trym Staurheim 22 Apr at 16:02	143	✓ succeeded
SK:#Added the same font size as in the other views... 42ff9f31 HA Henriette Andersen 22 Apr at 12:31	141	✓ succeeded
SK:#355 Added the right font size to both table 8c059cbc HA Henriette Andersen 22 Apr at 13:44	142	✓ succeeded
SK:#176 Made Kos Component. SK:#341 Kos can n... 9c1a03fb TS Trym Staurheim 20 Apr at 15:19	139	✓ succeeded

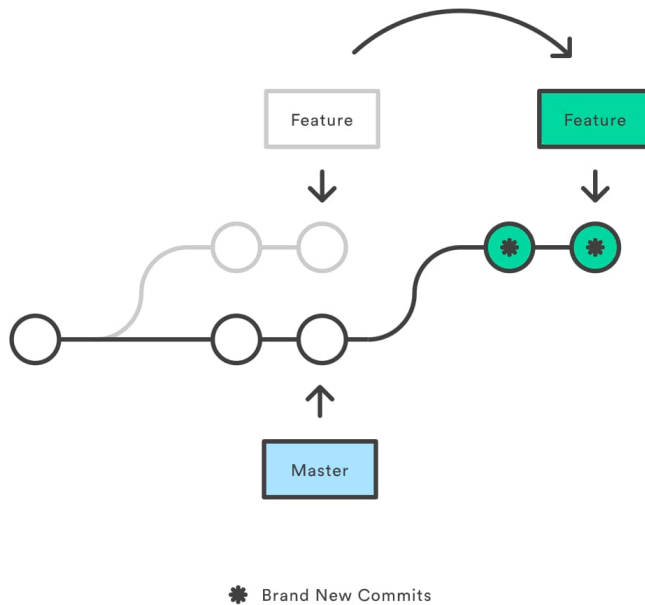


Figure 3: Git Flow

### Testing

To validate external quality, we created unit and integration tests; Unit tests to confirm that every method works in isolation, and integration tests to certify that methods work alongside their dependencies to provide the required correctness, completeness, and consistency.

We decided to focus on white-box and black-box testing in the backend and black-box testing in the frontend through end-to-end testing. White-box testing entails testing the internal structure of code, giving the tests direct access to private implementations. On the other hand, black-box testing entails that we test code without having access to its private and internal state, much like how a user would interact with a website (GeekForGeeks, 2020). In addition, we used code coverage, which provides us with raw statistical data on which files are covered by tests, providing overall control over the source code. Further tools used in testing will be covered in chapter 2.2.3.

### *Pair programming*

We applied pair programming to ensure that more than one developer knew the code, finding bugs and for security issues. As Linus Law says: “Given enough eyeballs, all bugs are shallow” (Wikipedia, 2021). Other benefits of using pair programming include (1) collective code ownership, as every team member gets more familiar with the code base, (2) facilitates for mentoring if there is a difference in competence level, and (3) better code quality because the existing and potential bugs are detected at an earlier stage.

### **2.2.3 Technological decisions**

Of our technological decisions, there are some tools we chose to use to increase our internal quality and particularly associated with maintainability. Firstly, we have chosen to use Project Lombok to reduce boilerplate code through annotations, creating and injecting setters, getters, and overrides on compile-time (Project Lombok, n.d.).

Secondly, we chose to use Spring’s implementation of Jakarta Persistence (formerly Java Persistence API, or JPA) to aid us with data access to our database, eliminating the need for us to write SQL queries ourselves. Thus, removing the possibility of human error from our side, effectively increasing the security, for example, removing the possibility of SQL injection.

Thirdly, we used Mockito, which is one of the most used stubbing tools for Java (Idan, 2016), to granulate the testing of components dependent on external parts of the application - such as methods dependent on the HTTP protocol, database connections, and service layers - which expedite internal quality by isolating the dependent parts of the system. Another tool we chose to use that is helpful for testing is the Zonky Embedded Database library. Zonky Embedded Database is a library that allows embedding PostgreSQL into Java application code with no external dependencies, allowing us to unit test with a "real" Postgres database (Zonkyio, 2021). We also used Flyway to facilitate database migration and versioning, allowing us to have a stable database with different versions of the main source code.

For our frontend development, we decided to use the CSS framework Tailwind to style our components instead of using traditional CSS files. Because it made it easier for us to style our React components, inserting class names directly into the markup. Another benefit of tailwind is that it allows for customization through its config file, here we added our own CSS classes,

which then gets compiled to usable CSS classes on compile-time, instead of having to create CSS files manually. See appendix 9 for tailwind config and example.

Lastly, the Product Owner challenged us to include some basic security in the application. To solve this, we chose to use the Spring Security Framework for password hashing and user authentication and Auth0 for Java Web Token generation and validation.

#### **2.2.4 Design principles**

Several decisions were made to ensure quality in the design. Some of them are implied by the requirements from Knowit. Considering the application is a replica, and the user group is already familiar with the old design. We have no means of affirming our design decisions with NKS; we decided to follow the Web Content Accessibility Guidelines (WCAG), Benyons design principles, and create wireframes. We also used the old system for reference to increase the overall familiarity of the system for the end-user.

##### *WCAG*

WCAG strives to achieve accessibility digital compliance. All businesses, organizations, and other entities who want their digital content accessible for all people should follow the WCAG guidelines (WCAG, 2020). Our supervisor from Knowit recommended that we should follow WCAG; nonetheless, all public websites and web content posted after January 1. 2012, must meet WCAG 2.0, which is law-enforced from January 1., 2021 (Kulturdepartementet, 2020).

##### *Benyon's Design Principles*

To facilitate quality in the design process, we decided to follow the 12 familiar design principles from Benyons book that cover *visibility, consistency, familiarity, affordance, navigation, control, feedback, constraints, flexibility, style, and conviviality* (Benyon, 2014, p.86). We chose to follow these principles because they are widely tested and recognized in designing a good user experience.

##### *Wireframes*

Wireframes provide a visual understanding of a web page during the early stages of the development process (Experienceux, n.d). We decided to create digital wireframes in the

initial phase of the process because we wanted to create an initial layout of the design and get confirmation and feedback from Knowit before proceeding with the implementation.

## **2.3 Project Management Choices**

Project management is a crucial part of succeeding in any project, and this aspect of system development has been a continuous focus over the past three years. In this project, we have held project management as a high priority. In the following sections, we will present our central decisions concerning project management, which includes the decisions regarding the use of Azure DevOps, Scrum, risk management, communication, and tools.

### **2.3.1 Azure DevOps**

We have used Azure DevOps as our central project management tool. Albeit Azure DevOps being a requirement from Knowit, it certainly would have been considered a highly relevant alternative with the characteristics of our project in mind. Other similar alternatives are Jira, BitBucket, or Trello, which hold some of the same features as Azure DevOps. The benefits of using Azure DevOps for our project include these: (1) DevOps facilitates working with agile methodologies such as Scrum, (2) it offers an end-to-end toolchain for developing and deploying software, (3) it offers a wide range of tools covering the entire development life-cycle with features such as boards, wikis, repos, and pipelines, and (4) it integrates with our selected IDE IntelliJ, as well as Git and Docker (DevOpsGroups, n.d.; StackShare, n.d.).

For project management, we have used these main features of Azure DevOps: Overview, Boards, Repos, and Pipelines. These features cover our Sprints, backlog, branches, pull requests, and documentation. It is important to mention that we have decided only to include the development part of the project in Azure DevOps in collaboration with our supervisor. Project management choices, writing on the bachelor's report, and other administrative tasks were left out of Azure DevOps. The reason for excluding administrative tasks in the Azure Board was that we considered them to oppose one of the main goals of the agile process: to deliver working software at the end of every sprint. It was also challenging to estimate the working hours of administrative tasks, as they were a continuous process throughout the whole project.



### **2.3.2 Scrum**

We have based our project management on Scrum, implementing the main elements of this methodology in our project. The elements we used included: Scrum Team, Daily Scrum, Product Backlog, Sprint Backlog, Sprint Planning, the Sprint, the Sprint Review, and the Sprint retrospective. These processes resulted in working software at the end of each sprint. We chose to use Scrum because the use of Azure DevOps facilitated an agile methodology, as well as us being familiar and having a positive experience with the use of Scrum in earlier projects.

We chose to set the duration of each sprint to two weeks, which led to us having a total of eight sprints. Setting the sprint limitation to two weeks was due to us having experience with three-week-sprints from an earlier project, and our impression was that we lacked a degree of consistent dialogue with the product owner and other stakeholders in the project. In contrast, we considered one-week sprints to be too short to provide valuable deliveries. We can also interpret that a one-week sprint would oppose some of the agile principles about trusting the team to get the job done and self-organize themselves (AgileManifesto, n.d.).

### **2.3.3 Risk Management**

In project management, it is essential to assess risks that could impact the product's success (Lavarya & Malarvizhi, 2008). To identify, evaluate, control, and mitigate potential risks, we chose to make a risk matrix (Appendix 3). We acquired an overview of risks, uncertainties, and threats in our project in the risk matrix. The content of the matrix implies what the risk is and what the consequence will be of each specific risk event. The matrix specified how we planned to manage the risk through proactive measures. The main reason for including the risk matrix was due to a strong recommendation from our supervisor at UiA, especially considering the unpredictable times of Covid-19. Secondly, this gave us a sense of control of which measures we had to implement in the project if Murphy's law were to take effect.

### **2.3.4 Backlog, Estimation & Priorities**

In Azure DevOps, we decided to use the entire Backlog hierarchy with Epics, Features, User Stories, and Tasks. An epic represents a business initiative to be accomplished; A feature typically represents a shippable component of software (Microsoft, 2021). In our case, there was a single Epic, being the whole application for NKS. Every view found within the

application makes up for one Feature each. These features are then split up into as many User Stories as are needed, which are subsequently split into Tasks. For every task, we made estimations of a best- and worst-case on how many hours we assumed it would take us, as well as an average of the two with a slight skew towards the worst-case estimation.

Work Item Type	Title
Epic	Sanitetskvinnene
Feature	Side for registrering av regnskap og møter
User Story	Som en bruker av Kvinners Sanitetsforening ønsker ...
Task	Designe regnskaps- og møteregistreringside
Task	lage regnskapstall component frontend
Task	Totale inntekter
Task	Totale kostnader
User Story	Som system ønsker jeg å ta imot spørringer og retu...
Task	lage controller, service, repository til regnskapstall
Task	Regnskapstabell DTO

**Figure 4:** Example of Epic-Feature-User Story-Task

To modularize the user stories and differentiate between the front-end and back-end tasks, we chose to make one user story based on the end-user requirements and one correlating user story based on the system requirements. These user stories were in turn connected to the overall feature/view. The reason for splitting the user stories into two was to make a clear differentiation between the system and the end-user as these have different requirements. It also made it easier for us to break down the tasks and thereby got a smaller scope for each task, as well as it made it easier to define the acceptance criteria of the task.

We estimated our tasks because it gives us greater control over the project in general and because we assume it to be a requirement due to Scrum's nature, as we need to know how many tasks we can put into one Sprint. It is also noteworthy that time is the only resource available; without time estimation, it would be difficult to plan or even prioritize the right features. For the individual Sprints, we can estimate how many tasks we can expect to complete. Estimating the backlog in its entirety can give us an overview of whether we will complete the project on time or not and facilitate prioritization through dialogue with the Product Owner.

In Azure DevOps, we also used the feature inside the Azure Board called “Capacity”. In the capacity section, one can assign daily hours to each team member to conduct different activities, including “development”, “design”, “documentation”, and “requirements”. As we decided to mainly use Azure DevOps for development purposes, we only set the capacity of the specific activities connected to the development process, namely “development” and “design”. We set the capacity of each team member to a maximum of four hours per working day. The capacity feature made it easier to structure and plan each day and helped us to set limitations for ourselves not to get overworked or underworked.

As mentioned in the *Characteristics of the Project*, most of the user stories found within the backlog are categorized as must-haves. In order to prioritize which user stories to work on, we decided to start with the ones that had the fewest dependencies. We were also advised to work in a “top-down” approach, where we create the frontend part of a view before creating the backend infrastructure that the view requires to function - an advice we chose to follow.

### **2.3.5 Communication**

During the project, we used several communication platforms to collaborate and maintain contact with our supervisors. The platform Microsoft Teams was used to communicate with our supervisor from Knowit, while the meetings with our supervisor from UiA occurred on Zoom. In instances where both supervisors were present, we used Microsoft Teams.

We used Discord, Messenger, and Google Drive for communication. The three platforms were used for various purposes, but were chosen based on our former experience of collaborating online. During the project, the main platform we used for formal communication within the team was Discord, primarily for online meetings, daily standup, and resource sharing. Discord made it easy to create an overview of different topics, both in regards to administrative tasks, but also directly relevant to development. Discord provides the opportunity to create channels based on different topics or paste formatted and syntax highlighted code straight in the chat. We used Messenger to plan the upcoming days, meetings, and other informal text communication. Google Drive and Azure Wiki covered all the administrative cooperation, including tables and relevant documents.

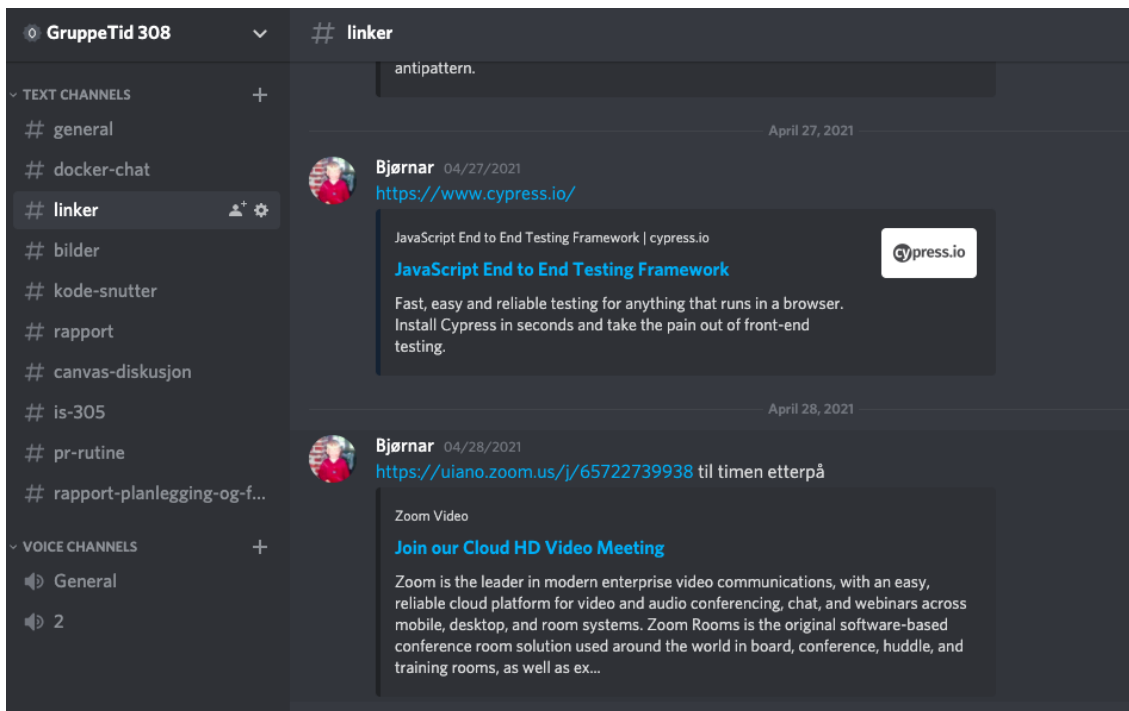


Figure 5: Discord window with different channels on the left and content of the selected channel on the right

### 2.3.6 Roles in the project

In the project's planning process, we made a group contract where we distributed overall roles to each team member. The roles included these: (1) integrator with responsibility for being a social contact person, (2) technical advisor with responsibility for giving supervision and being available regarding technical questions, (3) a secretary with focus on coordinating and having control over the administrative tasks, and (4) the primary contact person that was connected to our associative company Knowit. We chose these roles because some of us had experience with using the PAEI model before (Producer, Administrator, Entrepreneur, Integrator) and felt that these roles were well suited for project work (MindTools, n.d.). The reason for giving each person a role was to ensure that we considered every aspect of the project processes and ensure that every team member felt they were having an area of responsibility in the project. See Appendix 8 for the group contract.

In our project, we decided not to have a formal group leader. The reason for us not to designate any person the Scrum Master or leader role was with the intention of everyone feeling equally responsible for the project. We also followed the agile principle that the best architectures, requirements, and designs emerge from self-organizing teams (AgileManifesto, n.d.). This statement provided us with the flexibility of not choosing a leader because we were

confident that the roles already assigned to each team member were adequate and that the group democracy as a whole should pilot the project towards success.

## **2.4 Other central decisions in the project**

In the following sections, we will present central decisions not covered by the earlier subtopics in this chapter. We will present the decisions made in terms of role distribution in the system development part, as well as how and why we distributed roles in terms of the overall project.

### **2.4.1 Development Role Distribution**

At the beginning of the project, our team focused primarily on facilitating a productive and good process, especially emphasizing every team member to achieve an optimal learning outcome. In this project, we wanted to distribute most of the development tasks in the frontend, backend, databases, and documentation in a fair manner, so that every team member could:

1. Secure progression and get control of the project by familiarizing with every aspect of the system development. This decision was made to ensure quality and understanding of the project.
2. Ensure that every team member gets to learn different techniques, tools, and languages (e.g., Typescript, Java, PostgreSQL). This decision was made to fulfill the learning motivation of the team.

## **3.0 Running the project**

In this chapter, we will describe and present our approach to the project, including how we conducted planning and analysis, project management, agile methodology, design, and the development process.

### **3.1 Planning and Analysis**

The project began in January 2021 with the first Sprint. This Sprint became more similar to a pre-sprint, focusing on getting a clear description of the characteristics of the project and understanding the requirements. In this Sprint, we also focused on defining the project scope

in collaboration with our Product Owner. This Sprint also included the distribution of roles, installing necessary tools and software, and planning other administrative tasks.

### **3.1.1 Planning during Covid-19**

During the months we worked on this project, we needed to consider the global pandemic Covid-19. The shifting restrictions, the fear of getting infected, and ending up in quarantine or isolation became major factors in the planning of the project, as well as running the project. The first Sprint was mainly conducted through Discord as there was lockdown in Kristiansand with a strong recommendation of not visiting the University of Agder and a limitation of only two guests in our own home. The initial phase of a project, especially in regards to planning, was important for us to have face-to-face. The lack of meeting face-to-face presented a challenge for us as a team, as we all wanted to get adequate control over the project and the product. To cope with this, we had several and regular meetings using digital platforms with each other and our supervisor at both UiA and Knowit. Besides the project's planning phase, the pandemic has been a continuous challenge over the whole semester because of the ever-changing restrictions.

### **3.1.2 Planning administrative tasks**

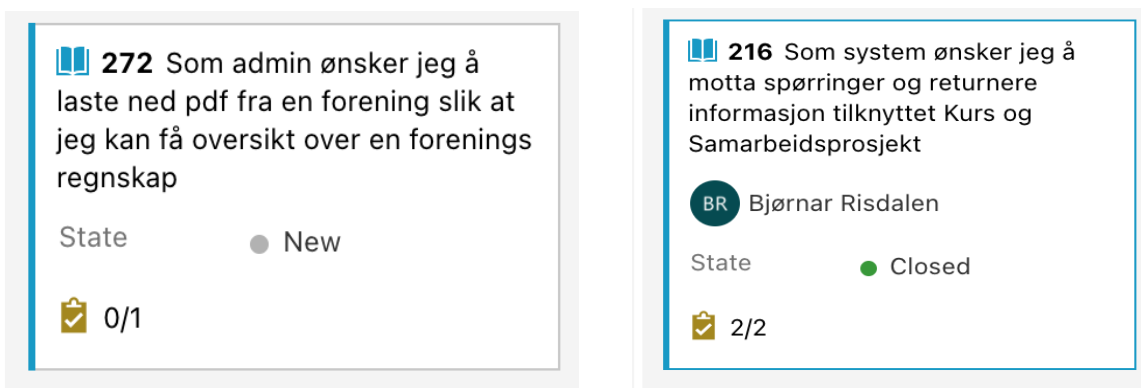
In the first Sprint, we were eager to get started with the development part of the project. However, we quickly realized that there was a bundle of administrative tasks that needed to be completed and planned before we could do any programming or development. The realization became apparent once Knowit held a kick-off at their offices in late January (Sprint 2). In this meeting, they provided us with helpful advice on how we should administrate and plan for tasks that were outside the development itself. They also recommended that we use about 60 percent of our time on development and the remaining 40 percent on administrative tasks. These factors made us discard most of the planning work we had done up until that point, and we decided to have a meeting where we planned the project in more detail and re-estimated and re-located both the development tasks and the administrative tasks. Subsequently, the re-planning increased the level of control and made the development process easier, driving the project forward.

### **3.1.3 Product Backlog**

A Product Backlog is an ordered list of the features and other activities a team may deliver to achieve a specific outcome (AgileAlliance, n.d). It was an appointed goal from both Knowit

and ourselves to set the Product Backlog as soon as possible in the project process. To establish the Product Backlog, we used different techniques, such as user stories and wireframes, and knowledge from Knowit to define the product backlog at an early stage.

In cooperation with Knowit, we decided to make one feature for each view in the application, with underlying user stories. Furthermore, we decided to make one user story for the end-user and one user story for the system. To ensure quality, we added acceptance criteria for each story, as well as estimating how much time each task in the user story would take based on the preliminary estimation. An example of two user stories (end-user and system) is presented in figure 6.



**Figure 6:** User stories

We used the MoSCoW (**M**ust have, **S**hould Have, **C**ould Have, **W**on't have) technique to prioritize the user stories. Each initial feature (view) was a must have, but we prioritized the features with the least external dependencies first. We added some additional user stories to the backlog throughout the project, such as fixing minor bugs and making small design adjustments. Some of these stories have been prioritized to be should have or could have.

### 3.2 Project Management

In this section, we will present how we organized the project management, including how we used Azure DevOps, how we conducted the risk management, the backlog and estimation, the burndown chart, communication, and how we distributed the roles in the project.

### **3.2.1 Azure DevOps**

We used Azure DevOps as our project management tool. We made use of the Azure Summary, Boards, Repo, and Pipeline overall features. In the Azure Summary, we used the Wiki to note advice, links, and other information regarding the project. We used the Azure Board to get an overview of the work items, the product backlog, and the sprint board. Azure boards helped us have control and progression in the project, as we could have an overview of each team member's capacity, which user stories with belonging tasks that were closed, and the estimation of each task. This feature also provided us with a burndown chart to predict our team's likelihood of completing the tasks within the allotted time frame. We could find both repositories and the associated files and the new, active, and closed pull requests in the Azure Repos. All the branches that had been used were also listed. We also used the Azure Pipeline to ensure we could build, test, and deploy software faster and easier. See appendix 10 for an overview of Azure.

### **3.2.2 Estimation of the backlog**

In the planning process, we made an initial backlog where we included each view of the application as a feature, with coupled user stories (split in two between end-user and system), and made associative tasks to them. When the tasks were completed, the user story was automatically put in the review tab in Azure DevOps and eventually closed. Since every initial user story was a must-have, the prioritization in our project was based on completing the views through the flow of the already existing system and how many external dependencies the view had, completing login first, and exporting to Excel last. We knew that it was important for Knowit that we completed one view 100 percent instead of 80 percent each, which was, therefore, an implicit prioritization. Throughout the semester, we found necessary tasks and user stories that we needed to add to the backlog, and these were prioritized based on dialogue with our supervisor and Product Owner Thomas, as well as through a discussion in the Development Team.

Estimation was a continuous main priority during the planning and running of the project. The subject IS-304 gave us a framework of how many weekly hours we should use to work with the project. IS-304 is a 20 credits subject, and it is therefore estimated approximately 25 hours a week per student, with a total of 810 hours during the whole semester per student. To ensure



that everyone in the team fulfilled this requirement, we were responsible for logging our daily/weekly/monthly hours.

In terms of estimating how many hours we would use on each user story, we sat down at the beginning of the semester to estimate each story based on experiences from earlier projects. After our kick-off with Knowit, they provided us with an Excel sheet they used in estimating tasks in their own projects, which was a valuable addition to structure the estimation. In this sheet, there were three columns that were split into worst-case and best-case scenarios for each user story and the associated task, where a formula calculated an estimation based on these scenarios. This estimation sheet also helped us divide the tasks, which resulted in completing a user story. In figure 7, we have provided a section of this estimation sheet.

A	B	C	D	E
Lokalforening				
<b>Task</b>	<b>Best case</b>	<b>Estimate</b>	<b>Worst case</b>	<b>Comments</b>
lage controller, service, repository til lokalforening	16	41	50	Må brytes opp
lage lokalforening component frontend	4	9	10	
lagre kall fra frontend til backend	4	10	12	axios; kall til hva da? Samme som "tidligere informasjon"
adgangsstyring	2	5	6	Lokalforening skal ikke kunne endre medlemstall
hente ut tidligere informasjon	3	7	8	axios
Lagre og gå videre	2	5	6	
Hente data i frontend-felter, sende som objekt	4	12	16	Sjekk f.eks. LokalForening og Login
Lage controller, service, repository for å ta imot DTO	5	15	20	Sjekk ForeningRepository og ForeningDao
Opprette DB migrering (SQL og flyway)	3	8	10	Kall fila V3_<timestamp>_lokalforening (ta kontakt om usikker)
Lagre DTO i DB	1	3.5	5	Sjekk ForeningRepository og ForeningDao
Total (hours):	43 h	112 h	138 h	
Estimate (weighting based on unsecurities)		114 h		

**Figure 7:** Estimation of Lokalforening component

When running the project, we experienced that we worked faster with the new views and that we could reuse the process and significant parts of the code in the later views. Partly because we were more experienced, but also because we had made central decisions regarding how we should structure the code, made universal components to use in views that required some of the same functionality as the first views, and that we had more clear roles in the project.

### 3.2.3 Burndown Chart

The burndown chart is a graphical representation of work left to do versus time and is often used in the agile work process to keep the team running on schedule and comparing the planned work against the team progression as well as monitoring the project scope creep (Visual Paradigm, n.d.). The burndown chart was a feature in Azure DevOps with different analytics, and contributed to us having an overview over the balance of remaining time and

tasks completed. This was very helpful for us to consider if we had too many tasks and user stories in each Sprint, if we were on schedule in terms of finishing the overall project, as well as the delivery of each sprint, helping us improve in the Scrum process. During the project, we improved in terms of analyzing and considering how many tasks were adequate to complete, and it has been a valuable tool throughout the semester to have control and ensure progression in the project.

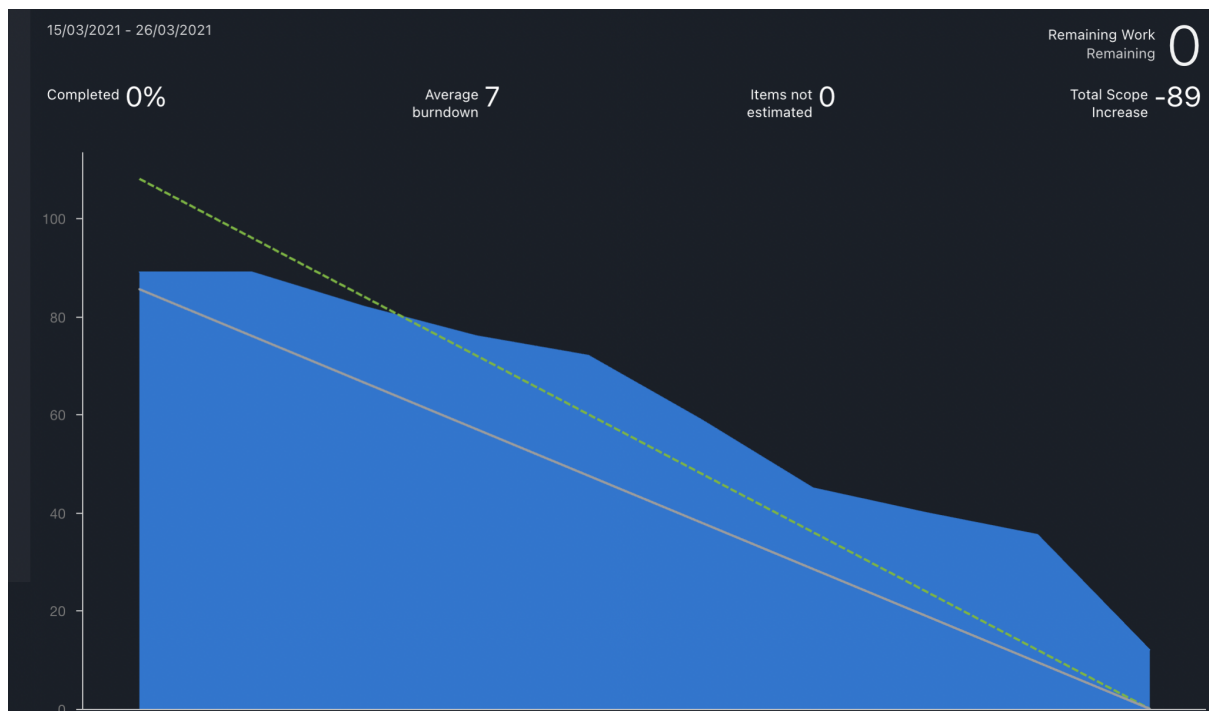


Figure 8: Burndown Chart

### 3.2.4 Risk Management

To cope with the potential risks in the project, we made a risk matrix. We used the risk matrix when some of the risks came into force, such as the pandemic forcing us to work from home. When the pandemic intensified, we knew that we had to be more structured in terms of clarifying planned appointments of when we should meet and how we could work to reduce the damage as this could potentially cause significant harm to our project. The fact that we had this overview made it possible for every team member to know which measures we should implement if unfortunate events were to happen. The overview has, in turn, helped us to maintain progression and control in the project. In figure 9, we have provided an extract of our risk matrix. The whole Risk Matrix is presented in appendix 3.

Risiko	1 til 5 Konsekvens	1 til 5 Sannsynlighet	Risiko	Tiltak for at dette ikke skal skje	Tiltak for å redusere skade
Gruppemedlemmer kan bli mindre effektive av å jobbe hjemmefra	4	3	12	Sette opp møter med faste klokkeslett. Jobbe i par for å motivere hverandre til enhver tid.	Noen må steppe opp å hjelpe og motivere hverandre.
Sent svar, eller lite oppfølging fra oppdragsgiver	4	2	8	Hyppig kontakt, avtale møter på forhånd.	Ta kontakt med veileder fra UiA, eller ta egne valg.
Manglende kompetanse i prosjektet	3	2	6	Gruppen setter av tid til å lese seg opp på nytt stoff.	Lære av hverandres styrker.

Figure 9: Risk Matrix

### 3.2.5 Communication

Communication has been a contributing factor for progress in the project, and to maintain quality in our work methods. Based on our previous semester, we were prepared for the collaboration to become more challenging than normal because of the pandemic. However, at the beginning of the semester, we planned early with our supervisor from UiA to meet up every second Wednesday if needed (on Zoom), and every second Monday with our supervisor from Knowit (on Teams). Communication has increased the quality of the project management, helping us remain in control. Based on our former years as a group, we have experienced that communication, in general, can be improved. Therefore our focus was to start planning early and lay a well-structured plan for the upcoming weeks and months.

At first, we planned that all members were to attend meetings physically, face-to-face on Monday, Tuesday, and Wednesday with flexi-time during 10:00-14:00. This seemed to work great until COVID-19 increased in Agder, and the University closed for a while. The pandemic did not have a big impact on our ways to communicate with each other, as we could use online methods to communicate. Unfortunately, the development process did slow down and became more time-consuming than before. The threshold for asking for help when we did not sit with each other physically became higher than before. It was not always optimal to use digital communication platforms because of the quality of screen sharing, microphone issues,

or other technical difficulties, affecting processes like pair programming, Daily Standup, and general motivation. However, later in the semester, when the University reopened, we were determined to meet up physically every day, perform Daily Standup, and we experienced that we became more productive when we met physically due to better communication. The physical meetings also improved the group dynamics and made it easier for us to learn from each other with pair programming.

### **3.2.6 Roles in the Project**

We divided overall roles in the project in our group contract, so that everyone should feel a responsibility for the project. These roles got amorphous throughout the project as everyone had shifting motivations at different points in the project. This led us to fill in the blanks for the other members through some trying pandemic times, especially with the role of the integrator and secretary.

In terms of the development process, we had rather loosely defined roles, as we all wanted to learn in every aspect of the development. Throughout the project, the roles became more defined in terms of who had the primary responsibility for the design, the frontend development, and the backend development. The role distribution was not planned, but due to us having to spend considerable parts of the semester at home, these roles were somehow unconsciously distributed based on interests and earlier experiences and knowledge regarding programming and developing.

## **3.3 Agile Methodology**

In this section, we will present how we used the agile methodology - and more specifically Scrum - to run our project. We will introduce and explain how we used the Scrum Team, the Sprint Planning, the Sprint Backlog, the Daily Standup, the Sprint Review, and the Sprint Retrospective in the following sections.

### **3.3.1 Scrum Team**

A Scrum Team is a collection of individuals working together to deliver the required product increments (Visual Paradigm, n.d.). Our Scrum Team consisted of the Development Team (four members) and the Product Owner. Since we decided not to have a Scrum Master, it was everyone's equal responsibility to make sure the Scrum process was followed, to arrange

Sprint Planning meetings, Sprint Retrospective meetings, and Sprint Review meetings, as well as being responsible for the burndown chart and to make decisions in regards of prioritizing the Sprint Backlog.

### **3.3.2 Sprint Planning**

Sprint Planning is the first event of the Scrum framework and initiates the whole Sprint. The Sprint Planning is conducted by the whole team and presents all work that is to be performed for the specific Sprint. The planning is about discussing the most important Product Backlog items and mapping them to the product goal. In addition, the team is able to invite other people to provide advice (Scrum, n.d).

Our Scrum Team implemented the Sprint Planning in a mix-up with Sprint Review and Sprint Retrospective, because this was the time we had access to discuss together with the Product Owner. During these meetings, we were able to ask questions and address important topics of a Sprint Planning event: (1) What do you think of the backlog?, (2) What do you think of the priorities, is it accurate? (3) What else should we keep in consideration?. These questions led to open conversations and provided us with guidance for the next Sprint.

### **3.3.3 Sprint Backlog**

The Sprint Backlog is a plan that is developed by and for developers. The Sprint Backlog contains all tasks that are supposed to be accomplished during the sprint to reach the Sprint Goal (Scrum, n.d).

During the project, we created the Sprint Backlog with Azure DevOps tools. The Sprint Backlog has been an essential feature in terms of getting an overview, division of tasks, and time management. The task board was determined from the Sprint Goal, and the tasks were created based on that. An example of a Sprint Goal: “Complete all views one by one”, based on the specific Sprint Goal, the task board was filled with all started views. The Sprint Backlog increased control and progress in the project. The Sprints were laid out so that each team member assigned themselves to a task; as shown in the figure of the Product Backlog, each task in “active” is named, and the estimated hours for each specific task are shown. In addition, each task card can be fulfilled with all hours completed and all hours remaining, and one can see the original estimate. All team members could keep track of the work items, and

it was visible at all times who was working on what, and when their tasks were in review, or when the task was completed.

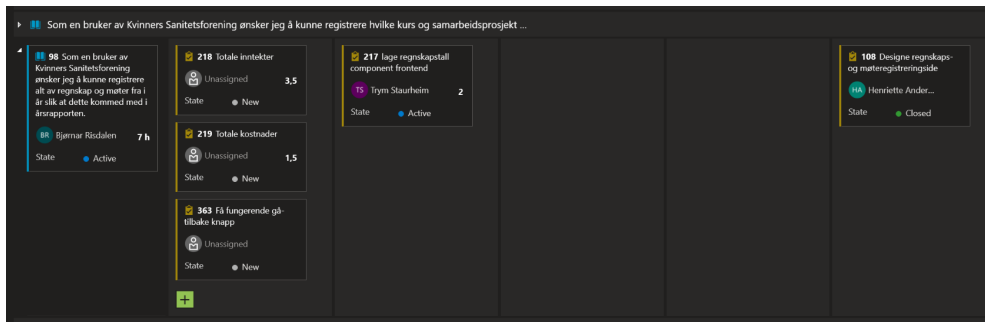


Figure 10: Sprint backlog

### 3.3.4 Daily Standup

The Scrum framework has named the Daily Standup “Daily Scrum”, but a “Daily Standup” is a time-boxed meeting where the team carries out a status check; the meeting is supposed to be held every day and at a set time (Agility, n.d). In our team, the Daily Standup was performed in a standard routine where each team members must stand up and answer the following questions:

- What have I accomplished since the last meeting?
- What do I plan to do for the next meeting?
- What impediments are in my way?

At the beginning of the project, we held daily standups two to three times a week until we understood the importance of performing them. Subsequently, we increased the amount of meetings to be held every day at a set time. The meetings lasted from five to ten minutes, and it was important for the team to focus on only answering the three questions briefly and to the point. It has been even more important during the pandemic to carry out the Daily Standup because the team has often been forced to work with the project from their home office. Therefore, the principle of Daily Standup could be considered an effort to increase the quality of the product because it facilitates communication and focus in the team.

### 3.3.5 Sprint Review

The Sprint Review meeting is one of the last events in the Scrum framework and includes these elements: (1) the attendance of the Scrum Team, the key stakeholders, and the Product Owner, (2) explanations on what Product Backlog items that have been completed, (3)

discussion of what went well during the sprint, problems during the sprints and how the problems were solved, as well as (4) a demonstration of the work that has been done (Scrum, n.d.).

Our team had Sprint Review meetings every other Monday; it was set for 1-2 hours per meeting. The Development Team and Product Owner attended all Sprint Reviews. For two of the Reviews, our supervisor from UiA was present as well. To get control over each meeting, our team created a table for each Sprint that contained “Sprint status”, “Things to Demo”, “Quick Updates”, and “What's next”. The Sprint Review meetings always ended with a completed Backlog for the next Sprint, and led to a fluid dialogue with significant inputs from our supervisors.

Sprint Status	<p><b>Where are we in terms of project completion?</b> Over halfway (sprint) , Stories completed: 7 , Views that are left: (aktiviteter, kos, regnskapstall og rapport)</p> <p><b>How many sprints left ?</b> Vi er nå i sprint 7, 2 Sprints left</p> <p><b>Is the project on track?</b> 71% completed user-stories (56% -&gt; 71%), Estimat: Eiendom og bidrag hadde 98 timer, men vi brukte 33 timer. (40), Styret brukte vi 19 timer (bedre enn best-case som var 24) og hadde estimert 64.</p> <p><b>Project completion % :</b> Project on budget? Project on track? Yes (but not on completing <i>everything</i>)</p> <p><b>Challenges:</b> Rapporten (fordelt timer på rapport og programmering), IS-305 innleveringer, Deadline (Både prosjekt og rapporten er deadline på likt)</p> <p><b>Sprint goal:</b> Fullføre den påbegynte viewsene, før vi startet på</p>
Things to Demo	<p>Test coverage gått opp (74%), Fjernet unødvendige filer fra test coverage, Slettet ubrukte metoder, All backend for EiendomVirksomhet.</p> <p><b>Login:</b>(usikker) Login nå med foreningsnummer istedenfor lagret ID, <b>AdminPanel:</b>Radio buttons, <b>Lokalforening:</b> Fjernet “antall”, Kun Admin tilgang til spesifikke felter, Henter ut data fra database og sende data, “Lagre” pusher videre til Styret, <b>Styret:</b> Footer - standardisert, Begynt overgang til ny Table component, Styling styret, Sende og motta data., <b>Eiendom og V.:</b> Table implementert med table component, <b>Gaver og bidrag:</b> Sende og motta data, full overgang til ny Table component for gjenbruk,</p>
Quick Update	<p>Vise backlog: Her skulle jeg visst hva som ikke ble ferdig, men etter en fin crunch på fredag til og med søndag ble alle stories og tasks for sprint 6 ferdig.</p>
What's next	<p><b>Tasks:</b> Feilhåndtering, Aktiviteter, Kos, Backend / Frontend aktiviteter , Kurs opplæring, samarbeidsprosjekt og Regnskap (Vise backlog)</p> <p><b>Sprint goal:</b> Bli ferdig med alle views som er oppsatt. Hvis vi får tid tar vi med rapport viewet også. Hovedprioritering ila. sprinten har vi tenkt er feilhåndtering og tilbakemelding på allerede eksisterende views.</p>

Questions for Product Owner: *What do you think about what's next? What about the priority? Is it accurate? What else should we keep in consideration?*

**Figure 11:** Sprint review form used for preparation (Mendez, 2015).

### 3.3.6 Sprint Retrospective

Sprint Retrospective is the last event of a Sprint and includes a plan on how to increase quality and effectiveness. Performing a Sprint Retrospective helps conclude the sprint in the best possible way (Scrum, n.d). After every Sprint, our team performed a Sprint Retrospective to identify the best possible ways to improve our effectiveness and the development process. In the Sprint Retrospect, we decided to follow the typical retrospect model that involves the following questions: (1) What worked well? (2) What went wrong? (3) What could be improved? After discussing these questions, our team considered new ways to increase product quality by improving the work process. To accomplish this, we needed to implement the improvements for the next Sprint.

Hva gikk bra ?	- Lærte masse nytt, godt samarbeid, gode arbeidsrutiner, godt læringsmiljø, god oversikt, høyere kvalitet på arbeidet, oppgaver blir ferdige
Hva gikk dårlig ?	- Sprint Review, Lav motivasjon midt i sprinten, for store oppgaver, at sprinten blir avsluttet før retrospekt
Hva kan forbedres?	- Opprette en mal til sprint review(forberede seg bedre til sprint review), be om hjelp tidligere, rapporten, huske å skrive timer, akseptansekriterier for brukerhistorier.

**Figure 12:** Sprint retrospective table from 01.03.2021

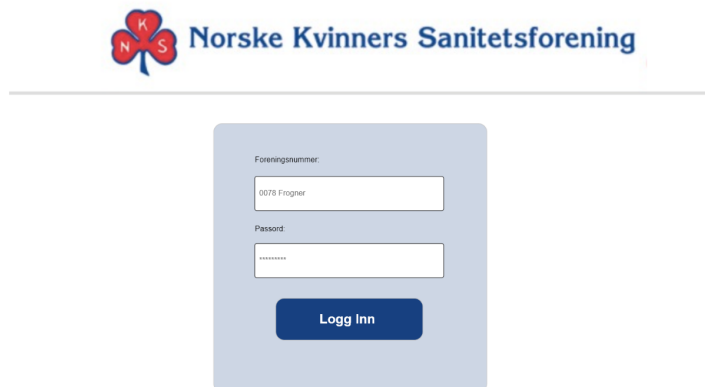
## 3.4 Design

The only requirement from Knowit concerning the design was to create a replica, but we were free to make changes in the design. We were of the opinion that the existing design of Sanitetskvinnene’s web page was outdated to some degree, and therefore we wanted to update the design and add a modern touch to it, through the use of WCAG, Benyon’s principles, and wireframes. Although we were able to make adjustments, there were still limited opportunities to make changes to the design. Based on the fact that the user group is an older target group, but also because they already are familiar with the previous design and most likely prefers it that way.



### 3.4.1 Wireframes

We made digital wireframes in the project's initial planning, and these were based on the user stories. The making of the wireframes contributed to us having a baseline in how each view should be designed. Creating wireframes was helpful for us as everyone got a common understanding of the design, and therefore we could argue that this increased the quality of the end-product and saved time at a later stage of the project.



**Figure 13:** Wireframe of the login page

### 3.4.2 WCAG

WCAG guidelines have contributed to increasing the quality of the design and making important design decisions. It has strengthened the quality of our product because it was a specific choice that is already tested, and the choices that we made all have a rationale behind them.

The principles from WCAG cover all standard design criteria, but not all principles were suitable for our design. Despite that, we did focus on universal design and worked on the principles that suited our design. Primarily, our focus was to work with the principles that could contribute to a better user experience for our user group. For example: Adding space between elements and text, add bigger font size, add headings, clearly show error messages, being responsive.

### 3.4.3 Benyons Design Principles

Following the design principles during the project was considered as an effort to increase quality in our design during the design process. The 12 principles are visibility, consistency,

familiarity, affordance, navigation, control, feedback, recovery, constraints, flexibility, style, and conviviality. The principles are categorized into three categories, learnability, effectiveness, and accommodation (Benyon, 2014, p.86-87).

The first principle is *visibility*, where the main focus is to ensure that things are visible; people need to see what the system is doing and the functions that are available (Benyon, 2014, p.86). Therefore we have consciously made the buttons bigger, and as before, we have clearly written their function inside of them. One example is on the login page that contains one button that should only perform one task, “logg inn”. The button is placed right under the input fields, where it is visible to the user.

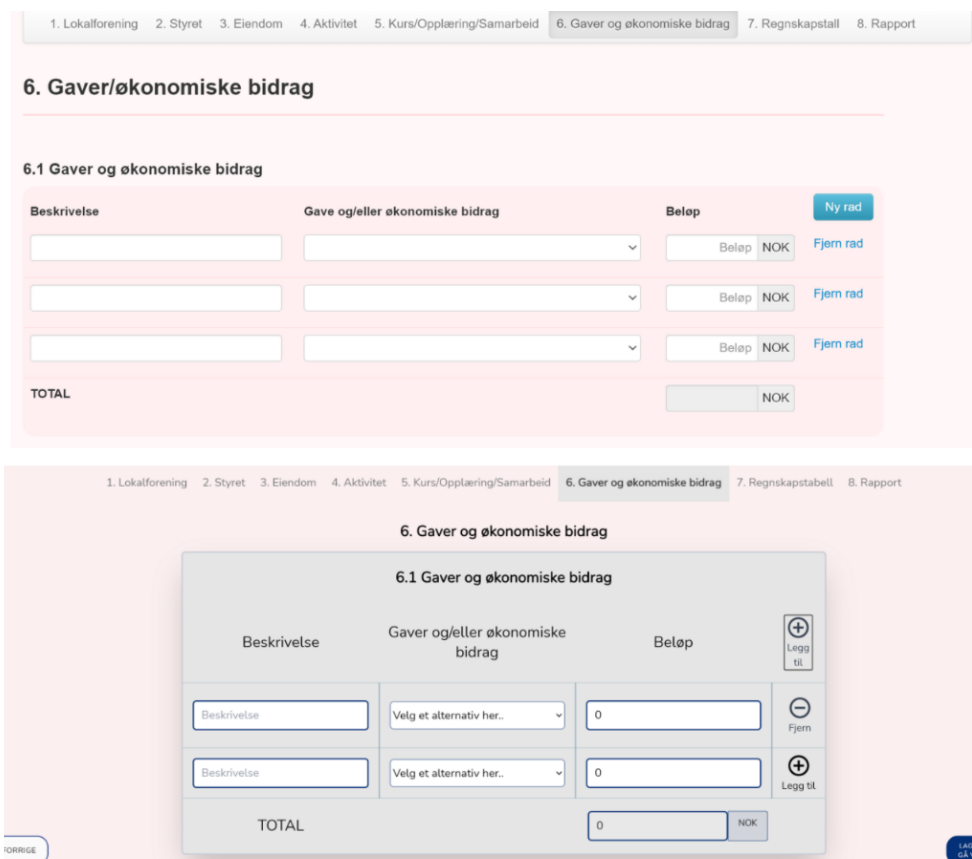


**Figure 14:** Example of visibility - existing version vs. our version

Secondly comes *consistency*. The principle focuses on being consistent in the use of design features and working with similar systems and standard ways of working. For example, be consistent in the use of colors, names, and layout (Benyon, 2014, p.86-87). Working around this principle, we kept focusing on creating the same layout as it was before to avoid

confusing the user. The design was already consistent in a way that all views kept the same layout, colors, and forms.

*Familiarity* is the third principle and focuses on using common symbols and language included (Benyon, 2014, p.86). Familiarity has been an important principle for us to follow as we were designing a replica of the old design. Our primary focus was to keep our user group familiar with the web page, such as using simple language to explain new activities. For example, in some of the views, we have an activity that says “legg til rad”. This is shown with two icons: a plus for adding and a minus for removing a row. In addition, we also explained the button with “Legg til” and “Fjern” to make sure that the user knows what the symbol’s activity is.

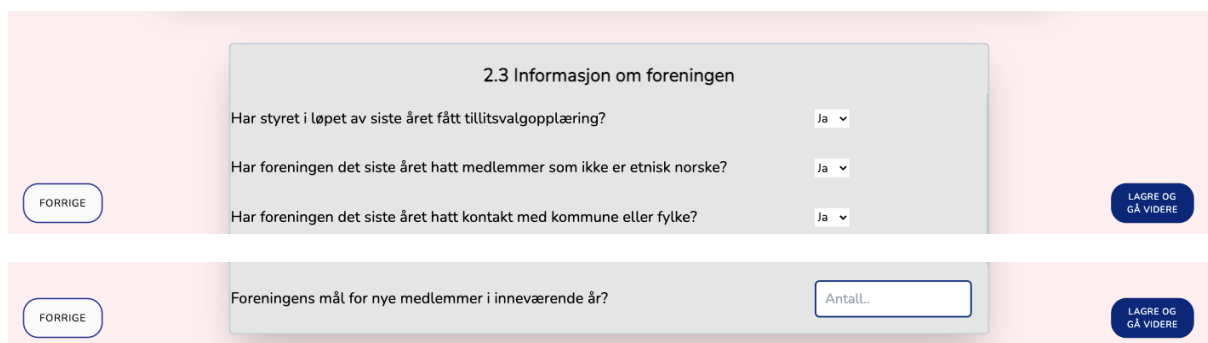


**Figure 15:** Example of familiarity existing vs. our version

*Affordance* is the principle that focuses on designing all elements, so it is clear what the element is. For example, when it comes to buttons, you have to design them to look like push buttons, to make people actually want to press them (Benyon, 2014, p.87). As we were to

keep the old design, we kept the shadow and hover effect on the buttons. This gave a clear sign of all the buttons and made them stand out from other elements on the page.

The fifth principle is *navigation* and is all about enabling the user to navigate through the system with directional signs, maps, and information signs (Benyons, 2014, p.87). The web page contained a navbar that made it easy to navigate forward and backward. We kept this as it was, since all points were presented clearly with the name of the next page. In addition, the webpage contained two buttons at the bottom of the site, one for navigating to the next view and one for navigating to the previous view. Through dialogue with our Product Owner, we were recommended to move the buttons into a footer that follows the webpage when scrolling - making it easier for the user to navigate.



**Figure 16:** Example of “sticky” footer

*Control* is the sixth principle; it must be visible who is in control and allow the user to take control (Benyons, 2014, p.87). The navigation bar is to help the user move around the web page but also to make them take control of their actions.

*Feedback* is principle seven and focuses on rapid feedback information from the system, so the user knows what effect their actions have. By applying constant feedback at the website will make the user feel in control (Benyon, 2014, p.87). The user will experience feedback in various forms throughout the website. For example, with the “Lagre og gå videre” button present that the adjustments that have been done are saved and the user can move on to the next page. Another example is the “Logg inn” button turning gray while the user is being authenticated.

The next principle is *recovery*, the principle that covers the importance of recovery from mistakes and errors. The user is supposed to be able to recover from their actions quickly and

effectively (Benyon, 2014, p.87). During the design process, we decided that feedback messages, in general, should be visible to the user; therefore, we decided to use a bright and visible color, “red”. We chose that color because it is an important message for the user. This way, the user is able to recover from their actions. Based on the background color we chose, which is light grey/blue, the recovery message appears visible since it is red.

*Constraints* are important to prevent the user from making errors or do inappropriate things (Benyon, 2014, p.87). The web page was already updated with constraints, and we kept it as it was because it was an important factor to prevent the user from making mistakes or doing inappropriate actions. The constraints on the site are that the user can only do changes that are meant for them. The input fields meant for admin are only available for the admin; if not, the input fields are disabled, so the user cannot write or click on them.



**Figure 17:** Example of the use of constraints

*Flexibility* is under the third category that should present the design “in a way that suits them” (Benyon, 2014, p.87). Flexibility is for people with different levels of experience and interest in the system. We developed flexibility as a small part of our design where the user is able to ‘Add a new row’ in a way that suits them. Experienced users can tabulate, and inexperienced users can use the mouse.

*Style* is principle eleven. The principle covers the main factor of the design's look; “It should be stylish and attractive” (Benyon, 2014, p.87). After guidance from the ten former principles mentioned, styling an attractive and stylish site became a lot easier. Our changes to styling include, but are not limited to: adjustments to make the design modern and attractive; adding a new background color that was inspired from Sanitetskvinnerenes main page; giving the buttons a much more modern touch with rounded corners (this was also inspired from their home page); we added icons that look like plus and minus were ‘add new row’ was; we

created a new look for the tables; lastly, we added a bigger font size and spacing between elements.

*Conviviality* covers that the interactive systems, in general, should be pleasant, polite,, and friendly. The system should not contain aggressive messages or abrupt interruptions (Benyon, 2014, p.87). Due to the last principle, we were focused on creating pleasant and polite feedback in the system. The importance of conviviality was highly prioritized when making choices regarding the design and how we have worded instructions and feedback messages. We considered it important that the user group could still understand and use it independent of the changes that were made.

The design principles have been a vital guidance throughout the process of making good design decisions.

### **3.5 Development Process**

In the following sections, we will present the running of the project regarding the use of technology, code review and code standard, the version control, the testing of the code, the CI/CD, and the use of pair programming to achieve a good development process.

#### **3.5.1 The Use of Technology**

All the technologies used have to some extent, impacted our development process. For example, being required to use Azure DevOps as a project management tool will impact parts of the development process, as all aspects in development are impacted by the sort of administrative tools used. However, here we wish to explicitly present how the different technological frameworks have impacted the development process in regards to facilitating internal quality in the form of structure and maintainability. We were already somewhat familiar with the Spring ecosystem and backend development in general; however, neither of us had much experience with the React framework, and Typescript was entirely new to us.

##### *Typescript*

To facilitate a healthy development process and later code efficiency, the group took an online course and read the Typescript handbook from the official documentation to understand the fundamentals of Typescript and how it impacts the structuring of code in React.

One of the primary benefits of Typescript is that it allows for optional strong static typing, which means that when a variable is declared in Typescript, it will only allow certain values to be assigned to it, and its type does not change (altexsoft, 2020). Compared to Javascript, which only interprets typing during runtime, Typescript enables errors and bugs to be caught before compile-time and can syntax highlight type errors in the code editor. Although Typescript requires more boilerplate code than Javascript to declare types for functions, objects, and classes, it is also efficient at type inference by dynamically inferring the type. Type inference has provided us with increased control making the code easier to maintain. Below is an example of how typescript infers a type and how to statically declare a type.

```
//type is inferred  
const hello = "Hello"  
  
//type is statically declared.  
let number: number
```

**Figure 18:** TypeScript type inference and type declaration

Continuing, we have used Typescript to create custom properties through interfaces, creating a boilerplate for what our components expect as property arguments for structuring our React components, making it easier to understand what types of arguments our components require. An example of an interface can be found in Appendix 3. The optional typing has been an incredibly helpful feature throughout the development process, as some types can be rather ambiguous in React's type system; explicitly allowing a type to have any type has helped us understand what the required type is by looking at the inferred type in the editor.

### *React*

The frontend was built using the javascript library React, which makes it easy to build component-based views for our application (Facebook Inc, n.d.). We used the popular create-react-app command-line tool to set up our frontend. Create-react-app includes all dependencies and setup needed, resulting in a ready-to-code skeleton (Facebook Inc, n.d.).

We used the composite pattern for our components, enabling us to reuse our components in different views of the application. For example, our input, buttons, and table components are generic, meaning they can be reused throughout the application, facilitating loose coupling

and high cohesion. Being able to create our own custom components complete with style and logic has made refactoring the different views that are dependent on them more straightforward to change, by allowing us to localize the change to the different custom components and tailor their implementation in their respective views. Midway through the development process, we had some difficulties with the GaverBidrag component's table, making the calculations, and correctly removing and adding rows to the state. Through some research and refactoring, we extracted the logic from the GaverBidrag component and made a separate generic Table component that could fit all our tables' needs, reducing code-duplication, as two other views need the exact same table and functionality. The refactored component reduced our codebase by a significant amount.

### *Spring/Java*

The Spring Framework has been a very helpful tool in our development process. With its implementation of Dependency Injection/Inversion of Control in its core systems (Spring, n.d.), Spring facilitates loose coupling through beans and autowired annotations. A bean is an object that is instantiated, assembled, and otherwise managed by a Spring IoC container (Spring, n.d.). We all have experience with creating a web application using Java from an earlier project, but using Spring simplified multiple points of the process like using controllers, services, and JPA repositories. The default maven integration also gives us easy access to additional tools, and the use of Spring Security enables username and password authentication through a single method override.

### *Tailwind CSS*

At the beginning of the project, we studied several different frameworks regarding CSS, from Material UI to Bootstrap to pure CSS. Initially, we chose to use Material UI, but we got a recommendation from a frontend developer to investigate Tailwind in more detail. After some research and experimentation, we chose to use this Tailwind CSS. The benefits of using Tailwind include these: (1) it increased the speed of the styling process because Tailwind helps you style the HTML elements directly, (2) it was possible for us to customize and make our own components, (3) we did not have to import the CSS files in our classes and files, and (4) it was convenient for us based on our former knowledge within system development, as Tailwind is class-oriented and provides built-in classes, which we are familiar with (Gebrewold, 2021). Overall we as a team have a positive experience with the use of Tailwind.



### **3.5.2 Code Review**

We used the team's standards for code and code review throughout the whole project. We used a fair amount of time and effort at the beginning of the project to agree on some common guidelines regarding code standards and the procedures in approving pull requests, the structure of the files and repositories (Appendix 2). The code was reviewed in Azure DevOps when one of the team members sent a pull request. At least two developers should review the code besides the author itself. If any team member had suggestions or discovered bugs or potential adjustments, we commented on that specific line of code (in Azure DevOps). The author would then make the adjustments to the code and would afterward amend the fixed code. Finally, the pull request was approved and eventually merged into main.

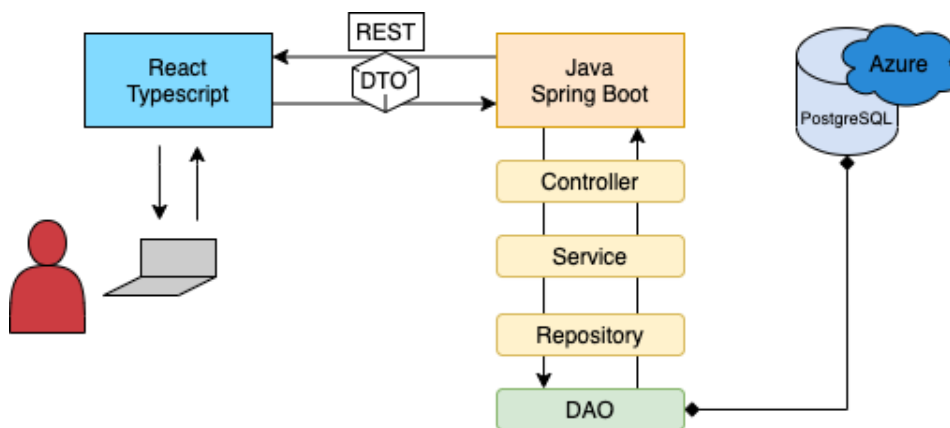
At the beginning of the project, we went through the review together physically by reading through the new and changed files in the PR, which facilitated unity during the code review process. These procedures ensured that every team member knew how to review the code, structure a file within the IDE, and other measurements regarding the code. As a result, we would argue that this has increased the quality of our source code, codebase, and the product itself. The code review and code standards are presented in Appendix 1 and 2.

### **3.5.3 Design Patterns**

When designing the codebase, we have followed Domain Driven Design (DDD), which entails the creation and categorizing of classes after what "objects" originate in the domain (Fowler, 2020). We used the old NKS application to find the specific domain objects. For example, every report is tied to a local Forening (en. Organization) - and so the very first class in our codebase was the Forening class, with an associated Forening table in the database. Iterating on this, we created Styret, Eiendom, Virksomhet, and other classes until the application was finished.

Overall, one can argue that the application is built around the model-view-controller pattern (MVC). In MVC, the Model usually reflects real-world things, the View is the code that directly interacts with the user, and the Controller receives user input from the View and decides how to manipulate the Models with that input (Codecademy, n.d.).

In our codebase, the entire React frontend is our View, the backend's Controller and Service classes make up the Controller, and our domain objects fetched from the database through the Repository classes are the Models. Our frontend displays forms and pages to the user, registers the input, and communicates with the backend using Data Transfer Objects (DTOs) using Representational State Transfer (REST). The Java/Spring backend receives the DTOs on specific endpoints registered with specific Controller objects. Figure 19 shows the application flow.



**Figure 19:** A model of the whole application

Following DDD and the Model and Controller part of MVC simplifies the process of creating new functionality in the backend and contributes to a higher internal quality by standardizing the coding process while also facilitating a higher degree of cohesion. First, we choose an object from the NKS domain, i.e., Eiendom. Then we create the Model/domain object in the backend with a corresponding database table before finally creating the Controller layer with controller and service classes. One could argue that we could achieve higher cohesion - as the service classes are responsible for too much - by creating classes like a ServiceSaver and ServiceUpdater. However, we figured this would decrease Readability as it could cause clutter in the project structure by creating classes with single methods, thus reducing the overall internal quality. We would rather have each service class responsible for its domain object.

We have also strived to achieve loose coupling in our code. For example, every service class depends on one or more repository classes for data fetching and object mapping. Our repository classes are created from interfaces and created at runtime with JPA, but one could implement that interface however one wants - i.e., with another library - and replace the JPA implementation with very few changes. The same thing can be said for our service classes in

general. Every controller is dependent on a service class; for example, `EiendomController` needs an `EiendomService` class. However, it only needs the service class to provide specific methods, like `getEiendommer` and `save`, so if needed, one could extend the `EiendomService` class and override its methods and register this new service class as the primary service class for Spring to use. We could have reached further loose coupling by changing all of our concrete implementations with interfaces and properly utilized Spring's dependency injection system, but we are overall happy with the degree of coupling and cohesion in our application.

### 3.5.4 Version Control

We used the Gitflow workflow in our version control. We decided to use Gitflow early in the project because it was both a recommendation from our Product Owner and we had some earlier experience with this workflow. It was important for us that every task in Azure DevOps had its dedicated branch (Feature Driven); the creation of branches included some standard guidelines and that no one should work directly in the main branch as this was the production-ready branch. Another recommendation from our Product Owner was that we rebased the commits before the pull requests were made. The significant benefit of rebasing, compared to merge, is that the project history is linear, making the commit history cleaner. It eliminates unnecessary merge commits, and it results in a linear project history where one can follow the tip of the `feature` all the way to the beginning of the project without any forks (Atlassian, n.d.). The joint guidelines included that each branch should have the keywords "SK" followed by the number of the task, for example, `SK-179-lokalforening-component`. These uniform measures have contributed to a structured version control and branch standard, which in turn has increased the quality of the version control management.

### 3.5.5 Code Testing

#### *Frontend testing*

For the frontend, we started using Enzyme and Jest but found them tedious to work with, as they require, in our experience, a fair amount of setup for a simple unit test. Luckily for us, we discovered Cypress. Cypress is an End-to-End testing framework that makes black-box testing easy by using a built-in web-scrapers that navigates through your application (Cypress, n.d.). Cypress enabled us to test our frontend components without worrying about the internal state of a component. i.e., Cypress runs tests the same way we have debugged and manually

tested the frontend - by clicking and typing. Cypress has increased the quality of our codebase by providing code coverage and has increased the overall control by pointing out which lines of code lack code coverage - we even discovered some unused components. The black box tests confirm the functionality of our application as one can watch cypress run the tests in the browser.

Testing of the frontend can be a tricky process as design, and functionality is often more adaptive to change. We continuously tested the design manually and ensured that calculations were handled correctly by the system. When designing isolated components in React, the internal state of the component is not directly accessible by the user, and as such, we argue that it is more fruitful to test that the data flows better in the application. Lacking a user group to test the application on Cypress has to some extent, filled that role, and gave us 81,7 percent code coverage in the frontend. See Appendix 11 for an example of a cypress test.

### *Backend testing*

For the backend, we created unit and integration tests using JUnit 5. We created unit tests for all methods found in both the controller and service classes through most of our project, as these exist to verify that everything works in isolation and are generally the quickest tests to make (STF, 2020). However, as the delivery date approached, we wanted to finish all views, and in order to save time, we decided to create larger integration tests that verified multiple methods and their dependencies at once. In our opinion, this did not decrease the internal quality as we already had thorough unit tests for other views, and the final views' methods were designed in the same way. Therefore, we found more value in integration tests that tested the whole flow from saving new elements, duplicate values, and the like instead of every piece in isolation. The time saved allowed us to finish the remaining views and still maintain the required 70% or greater test coverage.

### **3.5.6 CI/CD**

Automation with a build and deploy regime has been a requirement from Knowit and, as such, a central part of our development process. CI/CD is a method to frequently deliver apps to customers by introducing automation into the stages of app development through continuous integration, delivery, and deployment (Redhat, n.d.). In Azure DevOps, this can easily be integrated using Azure Pipelines.

We created one Pipeline for the backend and one for the frontend. The build and deployment steps are defined using YAML - a human-friendly data serialization standard for all programming languages (YAML, n.d.) - and are saved in `azure-pipelines.yml` files at the root level of each repository for automatic detection and integration. The pipeline files can be written manually in any editor, but Azure DevOps has integrated editors on the website where one can search and simply configure every step instead.

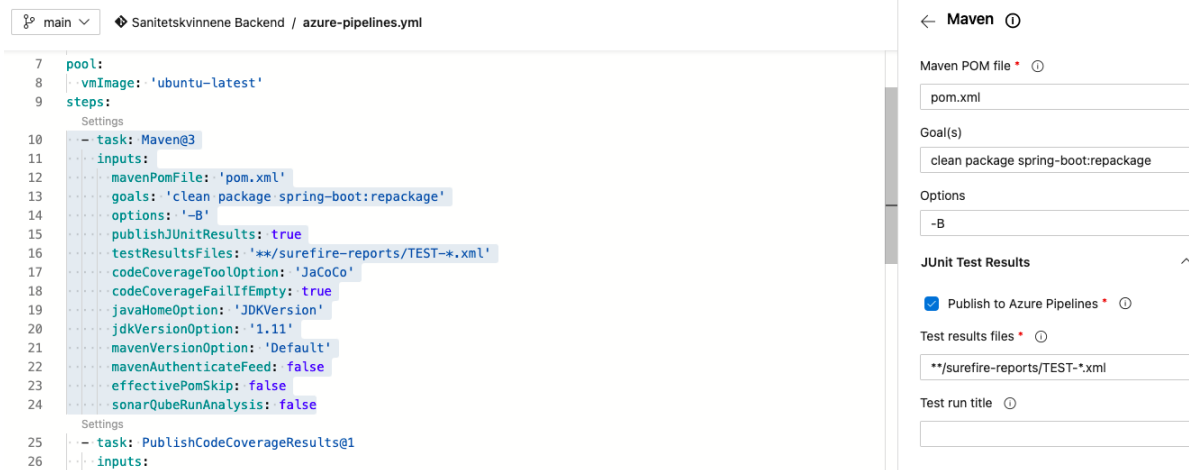


Figure 20: Pipeline YAML in the browser editor

Both pipelines have a trigger on their respective main branches from the git repository. This means that any change to the main branches triggers a pipeline run, which is what *Continuous Integration* entails. After the pipeline is started, the respective pipelines build the source code into runnable artifacts - the frontend using Node package manager and the backend using maven - after running and passing their corresponding tests. We consider this an important measure to maintain good external quality, as the pipeline will abort if *any* test fails and thus will never produce any erroneous software for the end-user as described by the tests.

The final task in our pipelines is the publishing of the generated artifacts. Using Azure resources, the artifacts are directly deployed to global servers, giving us access to the GUI, backend, and database on the web after every code change on the main branch - following the principles of *Continuous Deployment and Delivery*.

### **3.5.7 Pair Programming**

Pair programming is a technique that is used where two programmers work together. During pair programming, the developers change roles frequently, and the developers switch between writing the code while the other person observes (Wikipedia, 2020).

At the beginning of the project, we worked together at the University and performed pair programming, as usual, face-to-face. Our team conducted pair programming early in the process because we wanted to share knowledge and make sure that all team members got the same understanding of the system. Pair programming ensured that our team made greater progress in the project and maintained quality at a higher level. In addition, the use of pair programming increased the overall productivity of our team and facilitated increased structure in the code. Because one can detect errors earlier in the process and then avoid having to wait for the code review to detect errors, and additionally, it provides an extra quality check.

## **4.0 The product**

Below is the link to a video where we present our application.

[https://www.youtube.com/watch?v=AgA\\_KRGFvxk](https://www.youtube.com/watch?v=AgA_KRGFvxk)

## **5.0 Reflection**

In the following chapter, we will present our overall reflections regarding the process of the project, including project management, the quality and quality assurance of both the process and product, and challenges we have met during this semester.

### **5.1 Project Management**

In this section, we will present our reflections regarding the project management choices we made both in the initial phase of the project and running the project. We will specifically share our reflections regarding the agile methodology, risk management, the estimation of the tasks, and communication.

### **5.1.1 Agile Methodology**

The group has strived to follow the Scrum principles, and in summarization, we can say that we had an overall positive experience with this methodology. The reason is that we succeeded in delivering functionality after each Sprint, that we held Daily Scrum often, and that we completed both Sprint Review and Retrospect after each Sprint. These procedures have been both planned and documented, and some of the documentation is presented in appendix 4.

Even though we think we succeeded in most parts of Scrum, we also had some implications. This included us not having Daily Scrum every week in the initial phase of the project and, to some extent, out in the semester. After a few Sprints, we understood how valuable this procedure was, and we decided to conduct it every Monday, Wednesday, and Friday, due to us having other arrangements on the other days with other subjects and work. After a few Sprints, we also understood the importance of spending more time planning the Sprint Review with our Product Owner and using a great effort to discuss and consider which tasks to include in the Sprint Backlog and how we should structure our work during each Sprint. We solved this by setting aside two hours every Friday before a Sprint Review and Retrospect to discuss the last Sprint, as well as plan the next one. It was a significant improvement for us compared to the start, where we tended to lack some structure and plan for these events. It is also important to mention that Azure DevOps has facilitated for us to have a good experience with working agile and is a tool we strongly recommend for projects in the future. We have learned and improved our agile working method over the past semester, and we are satisfied with our agile process.

### **5.1.2 Risk management**

In these trying times, we have experienced that risk management has been of enormous importance. To handle potential threats and risks, we made a Risk Matrix to become aware of potential risks and know which measures to implement if something were to happen.

Throughout this semester, we have met challenges that regarded Covid-19 restrictions; team members felt demotivated because of the circumstances, different personal issues, and some issues connected to the work with the application itself. The fact that we were aware of the potential risks connected to these implications has helped us pick each other up through measures such as conversations, the possibility of working from home, and personal challenges as a section in the Daily Scrum and arranging social events outside of school. We

consider that we have all done our best to minimize the potential risks, and we have succeeded in reducing the damage that could potentially harm our project result.

### **5.1.3 Estimation and Prioritization**

In terms of estimation, this has been an ongoing challenge throughout the whole semester. In the initial phase of the project, we were unsure if we were to include administrative tasks in Azure DevOps and how much time we should put aside for these tasks. After the kick-off at Knowit, we got some recommendations that we should use about 60 percent of our total time to work on development tasks (programming, design, etc.). In the same meeting, we were also recommended to only include these development tasks in the board at DevOps. Additionally, we received an Excel sheet to help us structure each view into tasks and then estimate each task in the view.

After estimating each view, we experienced having too large tasks; therefore, we split these tasks into smaller tasks. The reason for doing this was that it made it easier for us to estimate, as it is more motivating to initiate a task that is estimated to 2 hours instead of 15 hours. This measurement ensured more control over the scope of each task, as well as it helped us estimate similar tasks afterward. Even though we carried out these measures, estimation was still a continuous challenge, as we have tended to overestimate tasks, with an example of estimating error handling on a view to five hours which only took us less than an hour. Estimation has been argued to be important and a challenge from both our Project Owner and supervisor at UiA over the semester, and this has also been our experience. Even though we have misestimated at times, we have had a positive experience with trying to estimate, and it has also been a motivating factor that we have finished most of the tasks in less time than estimated.

Prioritization has not been of great concern in the initial phase of the project because all user stories were must-haves. Nevertheless, we have had to prioritize the must-haves, where we decided to follow the flow of the application of which view to develop first. As a result, we chose to start on login, followed by the admin panel, and further implement the views from left to right. We have also added additional tasks and user stories throughout the project, such as error handling. The user stories and tasks that were added were also based on recommendations from our supervisor at Knowit. We also had to prioritize within each Sprint to decide which tasks we were to focus on getting finished first. In this process, the Sprint



Review, Sprint Planning, and Sprint Retrospective have been valuable in discussing and considering how we should prioritize as a team.

#### **5.1.4 Communication**

Communication has been a key in succeeding in this project, and we have found the communication regarding issues outside the project scope as the most important in keeping up our motivation and work ethics. The physical meetings between the group have been the most valuable because it includes rich communication and the feeling of support in another way than we have experienced over digital platforms such as Discord and Messenger. Every team member has felt lonely to some degree this semester, and it has been a prioritization to support each other through these times. Even though we have had periods where we were forced to sit at home and felt challenges with motivation and loneliness, we kept a continuous dialogue through digital platforms and physical meetings.

The consciousness surrounding the importance of communication in these pandemic times has facilitated the success of our project and has been a key factor in us utilizing a good process and delivering a good product.

### **5.2 Quality of End Product**

We wanted to achieve a high-quality system, which has been an important aspect for us throughout the whole project. The quality of our product is ensured by us accomplishing both the internal and external quality measures, which included requirements from both Knowit and Norske Sanitetskvinner.

For Knowit, high quality meant us satisfying the requirements of finishing all the views, having adequate test coverage of at least 70 percent, and modernizing the stack. To ensure that we met these requirements, we implemented tests, predefined code standards, and version control throughout the project. We have gotten continuous feedback from Knowit, and this feedback has been overall positive, with some minor suggestions for improvements, which we have prioritized and adapted to. Because of the arguments listed, we are confident that we have delivered an end product of good quality to the client, and it has been confirmed by our Product Owner several times.

The end-users required that the system offer the exact same functionality as the existing application and that it should be easy and understandable to use, especially because of the characteristics of this user group. We have put a lot of effort into designing the web application by following guidelines such as WCAG and other sources on good design. It would have been productive to perform end-user testing on this group, but completing an end-user test proved difficult, as we did not have access to Sanitetskvinnene, and the covid restrictions made it difficult to find people in the target group. We considered asking our grandparents to test the application, but none of them was available or willing to try. If we had the time and access to the end-user, it would have been valuable for us to ensure further quality by completing user tests. We are still confident that this system facilitates a good user experience.

### **5.3 Challenges**

In the following sections, we will present the challenges we have had throughout the project regarding Covid-19, cooperation with our client Knowit and implications regarding roles and role distributions.

#### **5.3.1 Covid-19**

The global pandemic Covid-19 has brought numerous challenges that have impacted motivations, workflow, and the overall structure of this course. The semester was supposed to be our opportunity to work at a company, build relationships, and experience what it is like at these companies, putting ourselves out there and potentially getting a job in return. The semester was instead filled with a lackluster of social interactions, coupled with Zoom and digital meetings. The fear of becoming infected, infecting others, the general worry about the future has without a doubt been a force to reckon with. Although concerning, we have gained insight into the remote work experience and ascertained new skills to overcome the challenges we have faced. The importance of trust, giving the team flexibility and means to customize workflow has allowed the individuals in GruppeTid to perform at their best during these troubling times.

The amount of COVID-19 infections increased early in the semester, already in February in Kristiansand, which led to us having to work from home. Early in the work process, it became difficult to find motivation for a new year of working at home. Because our group was

looking forward to starting working at Knowit sør's office at least once a week - being able to know how they work and have the opportunity to meet some of their employees. Luckily, the kick-off with Knowit was not canceled before the lockdown started, and we got to visit their office. During the kick-off, we saw their office, meeting rooms and met some of their employees. Attending the kick-off with Knowit did give us insight into what a day at their office was like and their work environment. In addition, we got to see how much help it was "that one time" we got to visit them and received guidance "in person" with two employees from their company.

After the meeting with Knowit, the pandemic restrictions intensified, and the group collaboration became digital for a short period. Regardless, working from home did not impact the progress or the productivity of our individual effort, but it instead affected the collaboration within our team as it became difficult with rich communication for this period. Facing these challenges, we made several countermeasures to solve them in the best possible way. First, we made a group contract to ensure that the group shall meet every Monday, Tuesday, and Wednesday (either physical or digital) to work together on the project. The purpose of creating a group contract was to secure the project in relation to maintain good routines during the pandemic and establish rules in case conflicts should arise. Secondly, we agreed to keep the social aspect in the group going and meet up once in a while to be social together.

The third factor that did contribute to dealing with the challenges during the pandemic was Daily Stand Up: it has increased quality of our product and facilitated for us to communicate with each other. The stand-ups led to good conversations within the group and contributed to our understanding and motivating each other more easily and providing input for each individual person in the team. We also experienced that daily standup naturally moved the conversation into aspects regarding planning and how to move forward with the product; overall daily standup has contributed to driving this project forward.

### **5.3.2 Roles and Role Distribution**

In the initial phase of the project, we had formal roles stated in the group contract, which included integrator, administrator, technical supervisor, and contact person with the client Knowit. The fact that our team has worked together on projects for about three years has contributed to us having experience in which roles we should include and who should have

which roles. We chose not to have a Scrum Master, and in retrospect, we think we could have included this solely for the arrangement of internal meetings in the Team. Nevertheless, we do not think that having a Scrum Master would have changed the result of either the process or product as we felt an equal responsibility to contribute to accomplishing a good agile work process.

During the development process, there were no clear roles initially as we wanted to learn and experience every aspect of the project. Nevertheless, the roles were crystallized throughout the project based on interests and experience. In retrospect, we think that we could have made the role distribution in the development even more precise so that every team member felt responsible for one specific aspect of the development. We could argue that this would have led to more motivation and affiliation to a specific feature. In opposition to this, we could still argue that this would have led us not to familiarize ourselves with the whole project and that this would have made us more vulnerable if something in the risk matrix were to happen. Overall we can say that our role distribution has been important in facilitating the development process.

## **5.4 Summarization**

In closing, we are well satisfied with both the process and the product, considering we met the requirements from Knowit, completed all the views, implemented design principles, and had adequate test coverage of the whole system. We also documented the process thoroughly during the whole semester, and we also accomplished an agile work process in a satisfying manner. In retrospect, we can say that we had some challenges, especially issues connected to the unpredictable situation of Covid-19, but all in all, we are well satisfied with our delivery in this bachelor's project.

## 6.0 Statement from Client

knowit

### Statement from Knowit Sør

Knowit Sør assigned the bachelor project team consisting of Trym Staurheim, Hanne Sjørnsen, Bjørnar Risdalen and Henriette Andersen. As we have experienced it, they have delivered in a timely and agile manner.

Thomas Wang followed the team through out the project with a minimum of bi-weekly sprint demos and conversations of the application.

#### **The project**

The assignment was to modernize our customer Sanitetskvinnene's solution for reporting sub-organizational data. They were to use our inhouse stack throughout the toolchain, entailing Azure DevOps for project management, Azure Cloud for deployment and our inhouse standard development stack consisting of Spring/Java, React/TypeScript, and PostgreSQL.

As we see it, these are most prominent areas where they excelled.

- The team acted autonomous and self-organized, they continuously improved their agile/scrum method, toolchain usage and delivery speed.
- They were to use our inhouse stack which contained languages, frameworks and systems which was new to them. This was something they quickly took advantage of rather than stumbled on.
- They have delivered a solution of acceptable quality which ticks all the required boxes and some extras.

---

Knowit Sør AS  
Vestre Strandgate 27-29  
4611  
knowit.no

Document ID:  
Version:  
Document Type:  
Information Class: Public

Author: Thomas André Wang  
Date: 2021.05.05

1

## **7.0 Self evaluation**

### **Henriette Andersen**

During this project I have gained a lot of new experience and learned much when it comes to team collaboration or getting acquainted with new programs/technologies, and how it is to cooperate with a company. At the beginning of the project I contributed through the initial planning and analysis process, where I had the main responsibility for risk management, and I learned a lot about how to manage risk and how much it can affect a project. Throughout the project, I have had the main responsibility for the design, which has suited me well, considering that I personally am interested in details and aesthetics. Further, I have attended and contributed through all sprint activities such as: Sprint planning, Sprint Review and Sprint retrospective. In addition, I have had a responsibility for writing on the report.

### **Bjørnar W. Risdalen**

During this project I have gained valuable insight in how it is to work on a real IT project. My biggest takeaway has been how important continuous estimation and re-estimation of tasks have been. Personally I am most satisfied with finally learning how to configure and set up CI/CD pipelines, which has had me very intrigued since the summer of 2020.

In the initial phase of the project I have contributed to planning and analysing the project. I have participated in all phases of the Scrum process, as well as writing on the report throughout. Additionally I edited the product video and did the voice-overs. Early on I worked full-stack along with my team members, but soon fell into a more backend oriented role where I was responsible for all things pipelines and azure as well as the backend codebase. However, I really enjoyed the tailwind workflow, so I have throughout been collaborating with whomever wanted additional input on how to design components using that framework.

Overall I am happy with the product we have created and the processes we have been through, as well as our collaboration with Knowit Sør, despite the pandemic situation.

**Hanne P. Sjursen**

During this project I have experienced working on a full-stack project in collaboration with an IT company and this has by far been the semester I have learnt the most, both on how I work in a team and about my own strengths and weaknesses. In the initial phase of the project, I contributed to planning and analysing the project and contributing to a successful agile work process. I have worked full-stack, including design, front-end and back-end on several of the views, both by myself and with the use of pair programming. Furthermore, I have attended all of the Sprint activities, and I have had a responsibility for writing the report. I hope and believe that I can make use of these valuable experiences in a future job, where I hopefully can combine IT, with my bachelor's degree in public health work.

**Trym E. Staurheim**

At the beginning of the project, I contributed to the initial planning and analysis of the project and had a full-stack developer role alongside my team. Nevertheless, as the process and project matured, I fell more into a frontend developer role, primarily responsible for implementing logic into the frontend views and components.

However, I have also been active in pair programming, and rubber-duck debugging and contributed to the report. I have been involved throughout the entire scrum process and took great interest in understanding Azure DevOps. I have garnered valuable experience in project management and development, from the scrum process to working full-stack. I am particularly pleased about learning more about Typescript and React and setting up end-to-end tests with Cypress.

In sum, I am proud of the product we have delivered and witnessing myself and my peers grow as our development process has matured by adapting to ever-changing circumstances from Covid-19 to computer breakdowns.

## References

Agile Alliance. (n.d.). Product Backlog. Retrieved from:

<https://www.agilealliance.org/glossary/backlog/>

Agile Manifesto. (n.d.). Manifesto for Agile Software Development. Retrieved from:

<http://agilemanifesto.org/>

Agility. im. (n.d.). What is a daily stand up?. Retrieved from:

<https://agility.im/frequent-agile-question/what-is-a-daily-stand-up/>

Albano, J. (2020, April 21). Design Patterns in the Spring Framework. Baeldung. Retrieved

May 04, 2021, from: <https://www.baeldung.com/spring-framework-design-patterns>

Atlassian. (n.d.). Gitflow Workflow. Retrieved 04 29, 2021, from:

<https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow#:~:text=The%20overall%20flow%20of%20Gitflow,merged%20into%20the%20develop%20branch>

Atlassian. (n.d.). Merging vs. Rebasing. Retrieved from 05 06, 2021, from:

<https://www.atlassian.com/git/tutorials/merging-vs-rebasing>

Barnes, D. J., & Kölling, M. (2017). *Objects First with Java: A Practical Introduction Using BlueJ* (6th ed.). Pearson Education Limited.

Benediktsson, O., Dalcher, D., & Thorbergsson, H. (2006, June). Comparison of software development life cycles: a multiproject experiment. *IEE Proceedings – Software*, 153(3), doi:87-101. 10.1049/ip-sen:20050061

Benyon, D. (2014). *Designing Interactive Systems - A comprehensive guide to HCI, UX and interaction design* (3 ed.). Harlow: Pearson Education Limited.

Codecademy. (n.d.). MVC: Model, View, Controller. App organization explained. Retrieved

from: <https://www.codecademy.com/articles/mvc>



Comley, C., Urban, E., Helen Huang, H., Coulter, D., Levy, R., Jacobs, M., Chase Wilson, C., Sherer, T., Danielson, S., & EE, K. (2021, January 22). What is Azure DevOps? Retrieved April 29, 2021, from:

<https://docs.microsoft.com/en-us/azure/devops/user-guide/what-is-azure-devops?view=azure-devops>

Cprime. (n.d.). What is Agile? What is Scrum? Retrieved April 23, 2021, from:

<https://www.cprime.com/resources/what-is-agile-what-is-scrum/>

Cypress. (n.d.). Testing has been broken for too long. We figured it was time to fix it.

Retrieved May 11, 2021, from: <https://www.cypress.io/how-it-works>

Det kongelige kulturdepartement. (2020). Endringer i likestillings- og diskrimineringsloven mv. (universell utforming av IKT-løsninger). Retrieved from:

<https://www.regjeringen.no/contentassets/28fcfab6e6a746a58cb38a07802d9d1f/no/pdfs/prp2020210141000dddpdfs.pdf>

DevOpsGroups. (n.d.). What is Azure DevOps? Retrieved from:

<https://www.devopsgroup.com/insights/resources/tutorials/all/what-is-azure-devops/>

Experienceux (n.d.). What is wireframing? Retrieved from:

<https://www.experienceux.co.uk/faqs/what-is-wireframing/>

Facebook Inc. (n.d.). React A JavaScript library for building user interfaces. Retrieved 05 11, 2021, from: <https://reactjs.org/>

Feiler, P., & Humphrey, W. (1993, February 26). Software Process Development and Enactment: Concepts and Definitions. *Proceedings of the Second International Conference on the Software Process-Continuous Software Process Improvement*, 147-158. doi:

10.1109/SPCON.1993.236824

Fowler, M. (2020, April 22). MartinFowler. Domain Driven Design. Retrieved from:

<https://martinfowler.com/bliki/DomainDrivenDesign.html>

Gebrewold, Y. (2021). Is Tailwind Really Worth it? . Retrieved from:

<https://javascript.plainenglish.io/is-tailwind-css-really-worth-using-1830a706231a>

GeekForGeeks. (2020, August 5). Differences between Black Box Testing vs White Box Testing. Retrieved May 10, 2021, from:

<https://www.geeksforgeeks.org/differences-between-black-box-testing-vs-white-box-testing/>

Idan, H. (2016, May 10). The Top 100 Java Libraries in 2016 – After Analyzing 47,251 Dependencies. Retrieved from:

<https://www.overops.com/blog/the-top-100-java-libraries-in-2016-after-analyzing-47251-dependencies/>

Jonnalagadda, G., Shafey, S., Lynah, W., Massetti, M., Maerten, P., Wieczorek, S., Landzaat, S., Probst, M., Schuster, A., Oxborough, C., Bains, R., & Bonser, M. (2017, July). Create value for your organisation with Agile Project Delivery. 1. Retrieved from:

<https://www.pwc.com/gx/en/actuarial-insurance-services/assets/agile-project-delivery-confidence.pdf>

Knowit. (2020). Innovative løsninger. Retrieved from:

<https://www.knowit.no/tjenester/solutions/>

Lavanya, N. & Malarvizhi, T. (2008). Risk analysis and management: a vital key to effective project management. Retrieved from:

<https://www.pmi.org/learning/library/risk-analysis-project-management-7070>

Mendez, J. (2015, July 31). Techniques for improving the Sprint review Scrum. Retrieved from:

<https://www.jesasmendez.ca/techniques-for-improving-the-sprint-review-in-scrum/>

Microsoft. (2021, March 15). Define features and epics. Retrieved from:

<https://docs.microsoft.com/en-us/azure/devops/boards/backlogs/define-features-epics?view=azure-devops&tabs=agile-process>

MindTools. (n.d.). The PAEI Model. Retrieved from <https://www.mindtools.com/pages/article/paei-model.htm>

Project Lombok. (n.d.). Project Lombok. Retrieved from: <https://projectlombok.org/>

PYPL. (2021, 9. April). *PYPL PopularitY of Programming Language*. Retrieved from: <https://pypl.github.io/PYPL.html>

React. (n.d.). React. Retrieved from: <https://reactjs.org/>

Facebook Inc. (n.d.). Create a New React App. Retrieved from: <https://reactjs.org/docs/create-a-new-react-app.html>

Redhat. (n.d.). DevOps. *What is CI/CD?* Retrieved from: <https://www.redhat.com/en/topics/devops/what-is-ci-cd>

STF. (2020, September 13). Unit Testing. Retrieved from: <https://softwaretestingfundamentals.com/unit-testing/>

Stack Overflow. (2018, January 28). Which Methodologies Do Developers Use? Stack Overflow. Retrieved April 23, 2021, from: <https://insights.stackoverflow.com/survey/2018#development-practices>

Stack Overflow. (2021, 13. April). Stack Overflow Trends. Retrieved from: <https://insights.stackoverflow.com/trends?tags=reactjs>

Scrum Org. (n.d.). What is a Sprint Backlog? Retrieved from: <https://www.scrum.org/resources/what-is-a-sprint-backlog>

Scrum Org. (n.d.). What is a Sprint Planning? Retrieved from: <https://www.scrum.org/resources/what-is-sprint-planning>

Scrum Org. (n.d.). What is a Sprint Retrospective? Retrieved from: <https://www.scrum.org/resources/what-is-a-sprint-retrospective>

Scrum Org. (n.d.). What is a Sprint Review? Retrieved from:

<https://www.scrum.org/resources/what-is-a-sprint-review>

Spring. (n.d.). Core technologies. Retrieved from:

<https://docs.spring.io/spring-framework/docs/current/reference/html/core.html>

StackShare. (n.d.). Azure DevOps. Retrieved from: <https://stackshare.io/azure-devops>

Tailwind. (n.d.). Rapidly build modern websites without ever leaving your HTML. Retrieved from: <https://tailwindcss.com/>

Talbot, J. (2018, July 31). What's right with risk matrices? Retrieved April 23, 2021, from:

<https://www.juliantalbot.com/post/2018/07/31/whats-right-with-risk-matrices>

Tiobe. (2021, April). *TIOBE Index for April 2021*. Retrieved from:

<https://tiobe.com/tiobe-index/>

Visual Paradigm. (n.d.). What is Burndown Chart in Scrum? Retrieved from:

<https://www.visual-paradigm.com/scrum/scrum-burndown-chart/>

Visual Paradigm. (n.d.). What is Scrum Team? - Scrum Guide. Retrieved from:

<https://www.visual-paradigm.com/scrum/what-is-scrum-team/>

WCAG. (2020, 7. January). Web Content Accessibility Guidelines - What is WCAG?

Retrieved from:

<https://www.essentialaccessibility.com/blog/web-content-accessibility-guidelines-wcag>

Wicksell, T. (2019, 16. October). Taylor Wicksell and Tom Gianos at SpringOne Platform 2019. Retrieved from: [https://www.youtube.com/watch?v=mln3\\_o6qlBo](https://www.youtube.com/watch?v=mln3_o6qlBo)

Wikipedia. (2021, 17. March). Linus's Law. Retrieved from:

[https://en.wikipedia.org/wiki/Linus%27s\\_law](https://en.wikipedia.org/wiki/Linus%27s_law)

Wikipedia. (2020, 18.December). Pair programming. Retrieved from:

[https://en.wikipedia.org/wiki/Pair\\_programming](https://en.wikipedia.org/wiki/Pair_programming)

YAML. (n.d.). The Official YAML Web Site. Retrieved from: <https://yaml.org/>

Zonkyio. (2021, 20. January). Zonky Embedded Postgres. Retrieved from:

<https://github.com/zonkyio/embedded-postgres>

## Appendix 1: Group contract

### Gruppekontrakt IS-304: Bacheloroppgave i informasjonssystemer

---

**Gruppenavn:** Gruppe 16 - GruppeTid

**Gruppens deltakere:**

- Andersen, Henriette
  - Tlf: 47672757
  - E-mail: henra17@uia.no
- Risdalen, W, Bjørnar
  - Tlf: 93286267
  - E-mail: bwendelborgrisdalen@gmail.com
- Sjursen, Hanne P.
  - Tlf: 90645901
  - E-mail: hanne\_sjursen@outlook.com
- Staurheim, Trym
  - Tlf: 95166509
  - E-mail: trymerlend@hotmail.no

**Skal jobbe sammen om følgende prosjekt:**

IS-304 - Bacheloroppgave i informasjonssystemer. Vi har blitt tildelt et prosjekt/oppgave fra bedriften knowit Sør som handler om å modernisere og forbedre en tjeneste for en av kundene til knowit. Dette skal gjøres gjennom forbedring av ulike deler av et system, fra databasen (NoSQL til SQL), til backend (forbedre Java-stacken), til frontend (forbedre designet ved hjelp av React).

**Mål:**

Gruppen ønsker å oppnå en høy karakter i dette emnet, fortrinnsvis A, samt tilegne seg teoretisk og praktisk kunnskap og kompetanse innenfor arbeid med digitalisering og teknologi for en bedrift (knowit) i Kristiansand.

**Arbeidsmengde og arbeidsmetodikk:**

Gruppen planlegger å møte til forelesninger, samt til oppsatte gruppetimer. Vi ønsker å bruke normert tid, 26 timer ukentlig per person, i emnet. Dersom det er behov for ytterligere arbeid vil det tilrettelegges for dette.

Vi har i samråd bestemt oss for å bruke rammeverket til Scrum for å gjennomføre bachelorprosjektet. Vi vil derfor jobbe iterativt i sprinter på 2 uker til å begynne med. Etter å ha tilegnet oss erfaring med tidsestimering av oppgaver o.l. vil vi påberope oss muligheten til å kunne endre tidsformatet til sprintene.

**Deltakelse:**

Vi forventer at samtlige gruppemedlemmer møtes til avtalt tid. Dersom andre forpliktelser skulle oppstå forventes det at dette blir informert om til gruppeleder innen rimelig tid. Vi forventer at alle bidrar like mye til sluttproduktet. Gruppen har blitt enig om at vi innfører fleksitid, dvs. at vi har noen timer i løpet av dagen alle må være tilstede på gruppearbeid. Resten av timene kan disponeres når og hvordan hver enkelt medlem av gruppen ønsker.

**Rollefordeling:**

Kontaktperson: Trym Staurheim

Koordinator/sekretær og ansvarsperson (frister, innlevering, presentasjon) : Hanne Sjursen

Teknisk veileder: Bjørnar W. Risdalen

Integrator (kontaktperson v/ konflikt, fokus på samarbeid i teamet o.l.): Henriette Andersen

**Samarbeidsverktøy:**

Discord, IntelliJ, PostgreSQL, Azure DevOps, Microsoft Azure, Microsoft Teams og Zoom.

**Regler:**

1. Respektere hverandres styrker og svakheter.
2. Respektere hverandres tid ved å møte innen rimelig tid med tanke på avsatt tidspunkt for jobb med bachelorprosjektet.
3. Gruppen skal bruke omtrent 26 timer pr. person til IS-304 - Bacheloroppgave i informasjonssystemer.
4. Alle på gruppen har plikt til å loggføre og holde seg oppdatert på Azure DevOps eller lignende.

5. Alle på gruppen må til enhver tid overholde gjeldende smittevernregler, f.eks. holde 1m avstand, gode hånd/hoste-rutiner osv.
6. Gruppemedlemmer skal være behjelpelig med å være ressurspersoner for de andre gruppemedlemmene dersom dette trengs. I begrepet ressursperson vil dette gjelde alt fra teknisk til praktisk til menneskelig bistand.
7. Gruppen vil anvende "strikes" dersom reglene blir brutt
  - a. (3) Personen får en muntlig advarsel.
  - b. (5) Personen får en skriftlig advarsel.
  - c. (7) Personen må forlate gruppen.

**Konflikt:**

Dersom det oppstår konflikt vil dette bli håndtert i 3 steg.

1. Først vil konflikten bli håndtert internt av ansvarlig for den sosiale kontakten i gruppen.
2. Neste steg vil være en konflikthåndtering i plenum mellom alle deltakere i gruppen.
3. Dersom de første stegene ikke fører til en endelig løsning, vil vi tilkalle veileder og eventuelt andre instanser ved UiA for å finne en løsning.

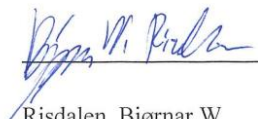
**Sted, dato:** Universitetet i Agder, 12.01.2021

**Underskrifter:**



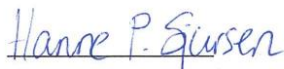
---

Andersen, Henriette



---

Risdalen, Bjørnar W.



---

Sjursen, Hanne P.



---

Staurheim, Trym E.



## Appendix 2: Wireframes

The wireframe shows the top section of a website for the Norwegian Women's Sanitary Association. At the top left is a logo consisting of a red four-leaf clover with the letters 'K', 'N', and 'S' on the leaves. To the right of the logo is the text 'Norske Kvinners Sanitetsforening'. In the top right corner, there is a greeting 'Velkommen, Admin!' and two buttons: 'HJEM' and 'LOGG UT'. Below the header, there are two main panels. The left panel is titled 'Finn forening her' and contains five search input fields labeled 'Søk på foreningsnummer eller navn:', 'Foreningsnummer:', 'Foreningsnavn:', 'Passord:', and 'Status:'. Each field has a magnifying glass icon and the text 'Søk...'. At the bottom of this panel is a link: 'Trykk "Årsrapport" for å eksportere i Excel ark fil.'. The right panel is titled 'Opprett forening her' and contains three search input fields labeled 'Foreningsnummer:', 'Foreningsnavn:', and 'Passord:'. Each field has a magnifying glass icon and the text 'Søk...'.

**Norske Kvinners Sanitetsforening**

Velkommen, Admin!

[HJEM](#) [LOGG UT](#)

### ^ Finn forening her

Søk på foreningsnummer eller navn:

Foreningsnummer:

Foreningsnavn:

Passord:

Status:

Trykk "Årsrapport" for å eksportere i Excel ark fil.

### ^ Opprett forening her

Foreningsnummer

Foreningsnavn:

Passord:

## 1. Lokalforening og medlemsinformasjon

### 1.1 Forening

Foreningsnummer:

700250 (Rapport sist levert: 2021)

Fylke

lorem ipsum

Organisasjonsnummer

lorem ipsum

Kontonummer

lorem ipsum

Facebook-referanse

lorem ipsum

Kommune

lorem ipsum

Lokalforeningens navn

lorem ipsum

E-post

lorem ipsum

Web-adresse

lorem ipsum

## 1.2 Medlemsinformasjon

Medlemsinformasjon legges inn av administrasjonen sentralt, og i

Antall medlemr

lorem ipsum

Antall betalende med

lorem ipsum

Antall nye medlemme

lorem ipsum

Antall utmeldte medlemm

lorem ipsum

Lagre og gå videre

## 2. Styret og informasjon om foreningen

### 2.1 Styret og informasjon om foreningen

Kopier fra forrige år

### 2.2 Styret i år (2021)

Har styret i løpet av siste året fått tillitsvalgopplæring?

Ja  Nei

Har styret i løpet av siste året fått tillitsvalgopplæring?

Ja  Nei

Har styret i løpet av siste året fått tillitsvalgopplæring?

Ja  Nei

Har styret i løpet av siste året fått tillitsvalgopplæring?

Ja  Nei

Forrige

Gå videre

## 3. Eiendom og virksomheter

### 3.1 Foreningen eier disse eiendommene

Navn /Adresse:	G. og bruksnummer	Type:	Ant boenheter:	Hvem er leietaker
...	300	300	...	300

Forrige

Gå videre

## 4. Aktivitet og inntektsgivende arbeid

### 4.1 Basis aktiviteter

#### Lorum ipsum?

#### Lorum ipsum?

Lorum ipsum?	Ja <input checked="" type="checkbox"/>	Nei <input type="checkbox"/>	Ja <input checked="" type="checkbox"/>	Nei <input type="checkbox"/>
Lorum ipsum?	Ja <input type="checkbox"/>	Nei <input type="checkbox"/>	Ja <input type="checkbox"/>	Nei <input type="checkbox"/>
Lorum ipsum?	Ja <input checked="" type="checkbox"/>	Nei <input type="checkbox"/>	Ja <input checked="" type="checkbox"/>	Nei <input type="checkbox"/>
Lorum ipsum?	Ja <input type="checkbox"/>	Nei <input checked="" type="checkbox"/>	Ja <input type="checkbox"/>	Nei <input checked="" type="checkbox"/>

Forrige

Gå videre

## 5. Kurs/opplæring og samarbeidsprosjekt

### 5.1 Deltakelse på kurs og opplæring i regi av N.K.S

Navn på arrangement:	Antall deltakere:
Frognerparken ..	300

### 5.2 Arrangement og prosjekt i samarbeid med andre

Navn på arrangement:	Samarbeidende forening:
Frognerparken ..	Skriv her..

Forrige

Gå videre

1. Lokalforening 2. Styret 3. Eiendom 4. Aktivitet 5. Kurs/oppl ring/samarbeid 6. Gaver og  konomiske bidrag 7. Regnskapstall 8. Rapport

## 6. Gaver og  konomiske bidrag

### 6.1 Gaver og  konomiske bidrag

Beskrivelse: Gave og/eller  konomiske bidrag: Bel p:

Frognerparken .. 300 300

Total:

Forrige G  videre

---

1. Lokalforening 2. Styret 3. Eiendom 4. Aktivitet 5. Kurs/oppl ring/samarbeid 6. Gaver og  konomiske bidrag 7. Regnskapstall 8. Rapport

Dersom du  nsker   skrive ut rapporten din, eller vil laste ned en PDF-variant av rapporten, velg "Last ned" under.

Last ned

### Appendix 3: Risk matrix

Lite sannsynlig:	1	Lite farlig:	1
Sannsynlig:	2	Farlig:	2
Meget sannsynlig:	3	Kritisk	3
Sv�rt sannsynlig:	4	Katastrofal	4

Risiko	1 til 5 Konsekvens	1 til 5 Sannsynlighet	Risiko	Tiltak for at dette ikke skal skje	Tiltak for � reduere skade
Gruppemedlemmer kan bli mindre effektive av � jobbe hjemmefra	4	3	12	Sette opp m�ter med faste klokkeslett. Jobbe sammen i par for � motivere hverandre til enhver tid.	Noen m� steppe opp � hjelpe � motivere hverandre.
At gruppemedlemme	3	4	12	Ha en klar plan p� hva som skal bli	Jobbe sammen p� store oppgaver

ne får for lite søvn				gjort hver dag.	som er mer tidkrevende.
For høyt stressnivå innad individuelt i gruppen.	3	4	12	Holde hverandre oppdaterte til enhver tid.	Gjennomføre daily stand up, slik at man har kontroll over prosjektet.
At et eller flere gruppemedlemmer kan bli smittet av covid-19	4	2	8	Ta hensyn, og følge smittevern reglene og restriksjonene	Lage et avsnitt i gruppe kontrakten som inneholder smittevern.
At gruppemedlemmer blir for lenge på campus	2	4	8	Gi hverandre en heads-up når man merker at produktiviteten synker.	Tiltak er å ha faste rutiner og klokkeslett.
Sent svar, eller lite oppfølging fra oppdragsgiver	4	2	8	Hyppig kontakt, avtale møter på forhånd.	Ta kontakt med veileder fra UIA, eller eventuelt ta egne valg.
Manglende kompetanse i prosjektet	3	2	6	Gruppen setter av tid til å lese seg opp på nytt stoff.	Lære av hverandres styrker.
Mangel på det sosiale i gruppen	2	3	6	Tillate å sette av tid til pauser, og til å diskutere andre ting enn prosjektet.	Ta jevnlig pause i 10-15 min i løpet av en arbeidstime.
At gruppemedlemmer glemmer å spise. F.eks Trym.	2	3	6	Minne hverandre på jevnlig pause med inntak av mat.	Ta med snacks som frister Trym.
Gnisninger i samarbeidet / individuelt	3	2	6	Opprettholde god kommunikasjon med gruppen. Forventningsavklaring. Holde et jevnt og godt samarbeid.	Opprette en god gruppekontrakt, og fordele roller.
At et eller flere gruppemedlemmer må holde seg hjemme pga vente karantene	2	3	6	Holde seg til en fast kohort, og ikke treffe alt for mange utenom gruppemedlemmer.	De som er igjen på skolen burde være tilgjengelig digitalt, hvis en / flere må ha hjemmekontor.
Datamaskin "kræsjer"	2	2	4	Jevnlig gå gjennom om det er noen gamle programmer som kan avinstalleres.	Eventuelt kjøpe ny PC.

## Appendix 4: Sprint review

### Sprint review - Model

Sprint Status	Things to Demo	Quick Updates	What's next
<p><b>Project Status:</b> What is the status of the project?</p> <p>Where are we in terms of project completion? - Stories com</p> <p>How many sprints left ?</p> <p>Is the project on track?</p> <p><b>Project completion % :</b> (Storie completed, stories forecasted)</p> <p>#Sprints left for completion</p> <p>Project on budget? y/n Project on track? y/n</p> <p><b>Sprint goal:</b> Why is it worthwhile to run the sprint? What should be achieved? For instance, address a risk, test an assumption, or deliver a feature.</p>	<p>In this section our team should list everything that is considered “Done”, this should be demonstrated by the development team during the sprint review meeting.</p> <p>(Tip: Keeping track of the sprint review, use a list and mark items after you are done showing it).</p>	<p>List all sprint backlog items considered not “Done”, to get feedback from the stakeholders.</p> <p>(Tip: Write notes under the sprint review meeting to keep track of the process).</p>	<p>In this section the team should list all everything that will be the sprint goal for the next sprint.</p> <p>(In this section it is important to exchange feedback)</p> <p>Ask open questions: - What do you think about what's next?</p> <p>-What about the priority? Is it accurate ?</p> <p>-What else should we keep in consideration?</p>

Sprint Status	Things to Demo	Quick Updates	What's next
<p><b>Project Status:</b> What is the status of the project?</p> <ul style="list-style-type: none"> <li>- closed 9 stories</li> <li>- half way</li> <li>- The views that are left: (styret, eiendom, aktiviteter, kos, regnskapstall og rapport)</li> </ul> <p>Where are we in terms of project completion?</p> <ul style="list-style-type: none"> <li>- half way</li> </ul> <p>How many sprints left?</p> <ul style="list-style-type: none"> <li>- 4 sprints left</li> </ul> <p>Is the project on track?</p> <ul style="list-style-type: none"> <li>- 51% completed of all stories</li> </ul> <p><b>Project completion % :</b> Stories Completed: 51% stories forecasted)</p> <p>Sprints left for completion: 4 sprints left</p> <p>Project on budget? Yes Project on track? Yes</p> <p><b>Sprint goal:</b> Why is it worthwhile to run the sprint? What should be achieved? For instance, address a risk, test an assumption, or deliver a feature.</p>	<p>Frontend:</p> <ol style="list-style-type: none"> <li>1. Admin Panel (du kan nå oppdatere en forening, du kan slette en forening, du kan gå til lokalforening)</li> <li>2. Lokalforening (Design, trykk på lagre gå videre, så blir dataen stubba)</li> <li>3. Gaver og bidrag. ( Legge til å fjerne rad)</li> </ol> <p>Backend:</p>	<p>List all sprint backlog items considered not "Done", to get feedback from the stakeholders.</p> <p>Trym viser backlog.</p> <p>Henriette can write notes.</p> <p><b>(Tip: Write notes under the sprint review meeting to keep track of the process).</b></p>	<p>In this section the team should list all everything that will be the sprint goal for the next sprint.</p> <p><b>Sprint goal for sprint 5:</b></p> <ul style="list-style-type: none"> <li>- Bli ferdig med Admin Panel, Lokal Forening. og Gaver Bidrag</li> <li>- Jobbe mot en bedre burn down</li> <li>- Fortsette med daily stand up</li> <li>- Sprint retrospective</li> <li>- Komme bedre i gang med rapporten</li> </ul> <p>(In this section it is important to exchange feedback)</p> <p>Ask open questions:</p> <ul style="list-style-type: none"> <li>- What do you think about what's next?</li> <li>-What about the priority? Is it accurate ?</li> <li>-What else should we keep in consideration?</li> </ul>

--	--	--	--

Sprint review 06.04

Sprint Status	Things to Demo	Quick Updates	What's next
<p><b>Project Status:</b> What is the status of the project?</p> <p>Where are we in terms of project completion?</p> <ul style="list-style-type: none"> <li>- Over halfway (sprint)</li> <li>- Stories completed: 4</li> <li>- Views that are left: (styret, eiendom, aktiviteter, kos, regnskapstall og rapport) Styling is done on Eiendom.</li> </ul> <p>How many sprints left ?</p> <ul style="list-style-type: none"> <li>- 3 Sprints left</li> </ul> <p>Is the project on track?</p> <ul style="list-style-type: none"> <li>- 56% completed user-stories</li> </ul> <p><b>Project completion % :</b></p> <p>Stories Completed:</p> <ul style="list-style-type: none"> <li>- 51% stories forecasted)</li> </ul> <p>Sprints left for completion:</p> <ul style="list-style-type: none"> <li>- 3 sprints left</li> </ul> <p>Project on budget?</p> <ul style="list-style-type: none"> <li>- Yes</li> </ul> <p>Project on track?</p> <ul style="list-style-type: none"> <li>- Yes (but no on completing <i>everything</i>)</li> </ul> <p><b>Challenges:</b></p>	<p>Frontend:</p> <ul style="list-style-type: none"> <li>- Admin (Generell Feedback)</li> <li>- Du må nå være innlogget for å tilgang</li> <li>- Eiendom og virksomheter (Styling, tables, legg til og fjern rad)</li> </ul> <p>Backend:</p> <ul style="list-style-type: none"> <li>- Lokalforening</li> <li>- Test coverage</li> <li>- Error håndtering</li> <li>- json structure på response</li> </ul>	<p>One of the team members Show the backlog with items considered not done.</p> <p>(Tip: Write notes under the sprint review meeting to keep track of the process).</p>	<p>In this section the team should list all everything that will be the sprint goal for the next sprint:</p> <ul style="list-style-type: none"> <li>- Mandag &amp; Onsdag. (Kl 11.00, daily standup)</li> <li>- Style Styret</li> <li>- Style Rapport</li> <li>- Gjøre ferdig gaver og bidrag</li> <li>- Skrive tester for backend</li> <li>- Skrive flere frontend tester</li> <li>- lage radio buttons AdminPanel</li> <li>-</li> </ul> <p>- What do you think about what's next?</p> <p>-What about the priority? Is it accurate ?</p> <p>-What else should we keep in consideration?</p> <p>- time management</p>



<ul style="list-style-type: none"> <li>- No sleep</li> <li>- Covid-19</li> <li>- Test coverage</li> <li>- Motivation</li> <li>- Bjørnar's computer crashed</li> </ul> <p><b>Sprint goal:</b> Why is it worthwhile to run the sprint? What should be achieved? For instance, address a risk, test an assumption, or deliver a feature.</p>			
---	--	--	--

#### Sprint review 16.04

Sprint Status	Things to Demo	Quick Updates	What's next
<p><b>Project Status:</b> What is the status of the project?</p> <p>Where are we in terms of project completion?</p> <ul style="list-style-type: none"> <li>- Over halfway (sprint)</li> <li>- Stories completed: 7</li> <li>- Views that are left: (aktiviteter, kos, regnskapstall og rapport)</li> </ul> <p>How many sprints left ?</p> <ul style="list-style-type: none"> <li>- Vi er nå i sprint 7, 2 Sprints left</li> </ul> <p>Is the project on track?</p> <ul style="list-style-type: none"> <li>- 71% completed user-stories (56% -&gt; 71%)</li> </ul>	<p>Frontend: Backend:</p> <ul style="list-style-type: none"> <li>Test coverage gått opp (74%)</li> <li>Fjernet unødvendige filer fra test coverage</li> <li>Slettet ubrukte metoder</li> <li>All backend for EiendomVirksomhet</li> </ul> <p><b>Demo-rekkefølge</b></p> <p><b>Login:</b> (usikker) Login nå med foreningsnummer istedenfor lagret ID</p> <p><b>AdminPanel:</b> radio buttons</p>	<p>One of the team members Show the backlog with items considered not done.</p> <p>Vise backlog: Her skulle jeg vist hva som ikke ble ferdig, men etter en fin crunch på fredag til og med søndag ble alle stories og tasks for sprint 6 ferdig. :)</p>	<p>In this section the team should list all everything that will be the sprint goal for the next sprint:</p> <p>Vise backloggen Tasks:</p> <ul style="list-style-type: none"> <li>- Feilhåndtering</li> <li>- Aktiviteter</li> <li>- Kos</li> <li>- Backend / Frontend aktiviteter</li> <li>- Kurs opplæring, samarbeidsprosjekt</li> <li>- Regnskap</li> </ul>

<p>Estimat: f.eks eiendom og bidrag hadde 98 timer, men vi brukte 33 timer. (40)</p> <p>Styret brukte vi 19 timer (bedre enn best-case som var 24) og hadde estimert 64.</p> <p><b>Project completion %</b> :Project on budget? - Yes Project on track? - Yes (but no on completing <i>everything</i>)</p> <p><b>Challenges:</b></p> <ul style="list-style-type: none"> <li>- Rapporten (fordelt timer på rapport og programmering)</li> <li>- IS-305 (innlevering en gang i uken de neste tre ukene)</li> <li>- Deadline (Både prosjekt og rapporten er deadline på likt)</li> </ul> <p><b>Sprint goal:</b> -Å fullføre den påbegynte viewsene, før vi startet på noe nytt. -Parprogrammering -Samarbeidet -Vi har hatt mer kontroll i forhold til prosjektet (Vært mer på skolen, mer samarbeid, fulgt opp daily standup oftere)</p>	<p><b>Lokalforening:</b> Fjernet “antall” Kun Admin tilgang til spesifikke felter. Henter ut data fra database og sende data “Lagre” pusher videre til Styret</p> <p><b>Styret</b> Footer - standardisert begynt overgang til ny Table component styling styret. sende og motta data.</p> <p><b>Eiendom og V.</b> Table implementert (hvis alt går som det skal) - Table component</p> <p><b>Gaver og bidrag</b> sende og motta data. full overgang til ny Table component for gjenbruk DELETE hvis fjerner en rad og sender på nytt</p>		<p>Sprint goal: Bli ferdig med alle views som er oppsatt. Hvis vi får tid tar vi med rapportviewet også. Hovedprioritering ila sprinten har vi tenkt er feilhåndtering og tilbakemelding på allerede eksisterende views.</p>
--	--	--	--

### Sprint Review 30.04

Sprint Status	Things to Demo	Quick Updates	What's next
---------------	----------------	---------------	-------------

<p><b>Project Status:</b> What is the status of the project?</p> <p>Where are we in terms of project completion?</p> <ul style="list-style-type: none"> <li>- Stories completed: 11</li> <li>- Views that are left: 0</li> <li>- En mangel i admin panel, der vi må knytte opp en knapp til backend</li> </ul> <p>How many sprints left ?</p> <ul style="list-style-type: none"> <li>- Vi er nå i sprint 8, siste sprinten</li> </ul> <p>Is the project on track?</p> <ul style="list-style-type: none"> <li>- 88% completed user-stories (71% -&gt; 88%)</li> </ul> <p><b>Project completion % :</b></p> <p>Project on track? Yes</p> <p><b>Challenges:</b></p> <ul style="list-style-type: none"> <li>- Estimering har vært både en utfordring og en forbedring fra forrige gang. Bra: 18t totalt, brukte 16t 15m Bommenet max 2 timer på en av oppgavene</li> <li>Utfordring: Bommenet med 3t 55 minutter på en task som tok 5 minutter.</li> </ul> <p><b>Sprint goal:</b></p> <ul style="list-style-type: none"> <li>- Bli ferdig med alle views som</li> </ul>	<p><b>Frontend:</b></p> <p><b>Backend:</b></p> <p>Test coverage gått opp (74% → 63,72%, men test for rapport generering lages i kveld)</p> <p>Rapport generes.</p> <p><b>Demo-rekkefølge</b></p> <ul style="list-style-type: none"> <li>- Vise feilhåndteringen på alle underveis</li> </ul> <p><b>ADMINPANEL:</b> fjernet at passord hentes ut</p> <p><b>Eiendom:</b> Fikset slik at bruker kan legge til rad mens de tabulerer.</p> <p><b>Aktivitet:</b></p> <ul style="list-style-type: none"> <li>- Vise hele Aktivitet</li> </ul> <p><b>Kos:</b></p> <ul style="list-style-type: none"> <li>- Vise hele Kos</li> </ul> <p><b>Gaver og bidrag:</b></p> <p><b>Regnskapstall:</b></p> <ul style="list-style-type: none"> <li>- Vise hele regnskapstabell</li> </ul> <p><b>Rapport:</b></p> <ul style="list-style-type: none"> <li>- Vise hele rapport</li> <li>- last ned Rapport</li> </ul>	<p>Ferdig med alle user stories i sprinten, utenom at års-rapporten i Excel er ~50%</p>	<p>In this section the team should list all everything that will be the sprint goal for the next sprint.</p> <p>I siste sprint:</p> <ul style="list-style-type: none"> <li>- Logg-ut knapp</li> <li>- Fix ups - footer, glemt passord, fikse tables på regnskapstabell</li> <li>- (Gjøre ferdig rapport)</li> <li>- Ferdigstille bacheloroppgaven</li> </ul> <p><b>Sprint goal:</b></p> <ul style="list-style-type: none"> <li>- Fullføre 100%</li> <li>- Trym har et personlig mål om å gå fra 10% -20%</li> <li>- Vi har satt oss noen delmål i forhold til rapporten osv</li> </ul> <p>Ask open questions:</p> <ul style="list-style-type: none"> <li>- What do you think about what's next?</li> <li>-What about the priority? Is it accurate?</li> <li>-What else should we keep in consideration?</li> </ul>
---	--	---	--

<p>er oppsatt. Hvis vi får tid tar vi med rapportviewet også.</p> <ul style="list-style-type: none"> <li>- Hovedprioritering ila. sprinten er feilhåndtering og tilbakemelding på allerede eksisterende views.</li> </ul>			
---	--	--	--

## Appendix 5: Sprint retrospective

Retrospective 15.02.2021

<p>Hva gikk bra ?</p>	<ul style="list-style-type: none"> <li>- Reestimerte backlog og fikk bedre oversikt.</li> <li>- Godt miljø</li> <li>- Kommet godt i gang.</li> <li>- God mappe- og kodestruktur.</li> <li>- Gode rutiner.</li> <li>- Fin commit history</li> <li>- Daily standup</li> <li>- Avtalt tacokveld</li> </ul>
<p>Hva gikk dårlig ?</p>	<ul style="list-style-type: none"> <li>- Dårlig estimering innledningsvis.</li> <li>- Dårlig til å skrive timer</li> <li>- Dårlig til å dokumentere</li> <li>- Litt lite produktive.</li> <li>- Sette av mer tid til retrospekt</li> <li>- Sette av mer tid til demo.</li> </ul>
	<ul style="list-style-type: none"> <li>- Bli flinkere til å opprette mindre tasks.</li> </ul>

Hva kan forbedres?	<ul style="list-style-type: none"> <li>- Bli flinkere til å føre opp timer</li> <li>- Bli til å dokumentere unnderveis;</li> <li>- Bli mer produktive</li> <li>- Daily standups kan bli mer spesifikk.</li> <li>- BLI ENIGE OM STANDUP RUTINE; mangel på struktur når standup.</li> </ul>
--------------------	---

#### Retrospective 01.03.2021

Hva gikk bra ?	<ul style="list-style-type: none"> <li>- Lærte masse nytt</li> <li>- Godt samarbeid</li> <li>- Gode arbeidsrutiner</li> <li>- Godt læringsmiljø</li> <li>- God oversikt</li> <li>- Høyere kvalitet på arbeidet</li> <li>- Oppgaver blir ferdige</li> </ul>
Hva gikk dårlig ?	<ul style="list-style-type: none"> <li>- Sprint Review</li> <li>- Lav motivasjon midt i sprinten</li> <li>- For store oppgaver</li> <li>- Sprinten blir avsluttet før retrospekt</li> </ul>
Hva kan forbedres?	<ul style="list-style-type: none"> <li>- Opprette en mal til sprint review(forberede seg bedre til sprint review)</li> <li>- Be om hjelp tidligere</li> <li>- Rapporten</li> <li>- Huske å skrive timer</li> <li>- Akseptanskriterier for brukerhistorier</li> </ul>

#### Retrospective 15.03.2021

Hva gikk bra ?	<ul style="list-style-type: none"> <li>- Sprint Review (opprettet en mal og fulgte den, god tilbakemelding)</li> <li>- Spørre om hjelp tidligere</li> <li>- Skrive mer på rapporten</li> <li>- Akseptanskriterier (Trym flink. kritikk til oss andre )</li> <li>- Størrelse på taskene gikk bra, men kan fremdeles forbedres</li> <li>- Generelt bra</li> </ul>
Hva gikk dårlig ?	<ul style="list-style-type: none"> <li>- Daily Stand Up</li> <li>- Hjemmekontor</li> <li>- Lavere motivasjon ifh til hjemmekontor</li> </ul>
	<ul style="list-style-type: none"> <li>- Påminne hverandre om å notere timer</li> </ul>

Hva kan forbedres?	<ul style="list-style-type: none"> <li>- Akseptansekriterier</li> <li>- Daily Stand Up ( kortere, presise, konkrete og fastsatte, annenhverdag )</li> <li>- Kutte ned på noen av taskene</li> </ul>
--------------------	---

#### Retrospective 30.03.2021

Hva gikk bra ?	<ul style="list-style-type: none"> <li>-Dele taskene inn i mindre oppgaver</li> <li>- Burndown var mye bedre</li> <li>- Time notering</li> <li>- Fleksibel capacity</li> <li>- Fulgte timene som var satt</li> <li>- For enkelte har det gått fint å skrive rapport</li> </ul>
Hva gikk dårlig ?	-Vagt med oppmøte på tiden
Hva kan forbedres?	-Rapportskriving (for noen)

#### Retrospective 16.04.2021

Hva gikk bra ?	<ul style="list-style-type: none"> <li>-Sprint Review (Demo,</li> <li>-Daily Stand up</li> <li>-Bedre rollefordeling</li> <li>-Skrevet mer på rapporten</li> <li>-Størrelse på taskene</li> <li>-Lagt ned litt ekstra timer i prosjektet</li> <li>-At vi kan samarbeide på skolen</li> <li>-Gruppen er generelt mer motiverte</li> </ul>
Hva gikk dårlig ?	<ul style="list-style-type: none"> <li>- Motivasjonen er opp og ned</li> <li>-Slurvefeil</li> <li>-Datastruktur kommunikasjonsfeil</li> </ul>
Hva kan forbedres?	<ul style="list-style-type: none"> <li>-Kommunikasjon</li> <li>-Backloggen, gjør den klar før vi faktisk starter på sprinten</li> </ul>

#### Retrospective 30.april

	-Sprinten var veldig godt planlagt og gjort helt klar før vi startet
--	--

Hva gikk bra ?	-Sprint Review (alle bidro) -Bra struktur på rapporten -Daily standup -Strls på taskene -Alle har gitt ekstra timer
Hva gikk dårlig ?	-Stress -Møte på tiden -Litt slurvefeil
Hva kan forbedres?	-Review av Pull request -Møte på tiden / Bedre kommunikasjon -Forbedre fordeling av oppgaver / punkter i forhold til rapporten

## Appendix 6: Code standards

The purpose of this document is to assert quality of source code through formalizing standards for the group to follow when writing code.

### General guidelines

- Use code formatting to keep code consistent. Use 4 spaces (one tab).
- No uncommented code.
- No unused code. Remove code that serves no functionality to the final product.

### Variables and Classes (Components)

- Classes, Interfaces and Components
  - Use Descriptive nouns for naming.
  - Mixed casing, starting with Capitalization of the first letter. Followed by Camelcasing (e.g., AdminPanelController).
  - Interfaces should start with a capitalized I to signify to the reader that it is an interface. (e.g., IMyInterface { name: string })
- Variable
  - Use Descriptive nouns for naming
  - CamelCasing. (e.g., foreningsNummer)
  - No use of one letter variables. Except for loops. (e.g., int x = 0;)
  - Avoid redundant variables.
- Functions and Methods

- Use Descriptive naming for what the method does. (e.g., sumIncome())
- Each method should be responsible for one action.

## Code structure

- Frontend
  - All components are placed in a folder with the same naming, with an adhering types file.
    - The type file must include the components name and “types” suffix.
  - All components should have an interface declaring prop types.
  - Utility functions and methods are placed in the utility folder.
  - Tests are structured with a respective folder for the component/method being tested.
- Backend
  - All classes are placed in accordance with the Repository pattern.
    - Repository classes are to be placed in the repository folder.
    - Domain objects are to placed in repositories’ subfolder domain
    - Controller classes are to be placed in rests’ subfolder controllers.
    - Service classes are to be placed in the services folder.
    - Data transfer objects (dto) are to be placed in rests’ subfolder dto.
    - Utility classes are placed in utilities.
    - Security classes are placed in security.
  - Tests
    - Unit tests follow the same structure as classes.
    - Integration tests are placed in the folder integration.
  - Migration files
    - are placed in the migration sub directory under resources.
    - Migration files are named: Vnumber\_date/time\_ *table/change.sql*
      - E.g., V3\_29041911\_alter\_table\_lokalforening.sql



## Appendix 7: Git Procedures

### Project Guide to Git and Version Control

In general:

- Each task in Azure DevOps should be a separate branch in Git
- No group member should work directly in the main branch
- The code is ready to be merged with the main branch when:
  - Functions in the code match the user story
  - Typo mistakes are removed
  - TODO or NOTE notations are removed
  - Unnecessary comments are removed
  - Print statements are removed
  - Check that formatting is right according to code standard
  - The code is written according to project code standard
  - The code has been approved by at least two required reviewers outside the author

## Git Procedures to Follow on Common Git Operations:

1. Push to own branch
  - o `git branch` - check that you are in your own branch
  - o `git status`
  - o `git add .`
  - o `git commit -m "SK:#32 descriptive text"`
  - o `git push`
  - o create pull request → code shall enter code review
  - o NB! Set up required/optional reviewers
2. How to fix up code according to feedback:
  - o go to your branch
  - o `git status`
  - o `git log`
  - o make changes
  - o `git add .`
  - o `git commit -amend`
  - o `git push -f`
3. Fix up in commits before merging so that only one commit appears in the log
  - o `git checkout main`
  - o `git pull main` (pull newest version of main)
  - o `git checkout own branch`
  - o `git rebase -i master`
  - o squash, reword, pick, fixup, drop the commits
  - o `git rebase -continue`
  - o If you encounter merge conflicts:
4. Fix up merge conflicts:
  - o Open the project, right click, and choose to resolve Git conflicts
  - o Doubleclick on conflicts to view conflicts. Choose left, right, or middle option depending on which version you choose to keep.
  - o Continue to all conflicts are solved
  - o When all conflicts are solved, run tests
5. Make branch ready to merge
  - o `git push -f`
  - o `merge`
6. Fix a bug
  - o make a fix-up branch that specifies the bug
  - o fix the bug
  - o `git add .`
  - o `git commit -m "SK:#23 descriptive text of the fix-up"`
  - o (if needed: `git rebase -i main`)
  - o `git push "branch name"`
  - o create pull request

## Appendix 8: Structuring of component interfaces.

Here is an example for one of our React components interfaces. It shows the function components arguments names and types making it easier for the team to know what the required arguments for this component.

```
export interface IadministrateForeningPanelProps {
  forening: Forening,
  handleForeningChange?: (event: React.ChangeEvent<HTMLInputElement>) => void
  setModal: React.Dispatch<React.SetStateAction<boolean>>
  message: React.Dispatch<React.SetStateAction<{ body: ResponseData }>>
  setCurrentForeningReportNumber: React.Dispatch<React.SetStateAction<number>>
}
```

Here is the interface implemented into the function component.

```
const AdministrateForeningPanel: FC<IadministrateForeningPanelProps>
```

```
const AdministrateForeningPanel: FC<IadministrateForeningPanelProps> = ({
  forening : Forening ,
  handleForeningChange : ... ,
  setModal,
  message,
  setCurrentForeningReportNumber
}) => {
```

## Appendix 9: Tailwind config file and example

```
module.exports = {
  purge: ['./src/**/*.{js,jsx,ts,tsx}', './public/index.html'],
  darkMode: false, // or 'media' or 'class'
  theme: {
    fontFamily: {
      'sans': ['Nunito']
    },
    extend: {
      keyframes: {...},
      animation: {
        'fade-in-down': 'fade-in-down 0.5s ease-out',
        'fade-out-down': 'fade-out-down 0.5s ease-out',
        'fade-in-up': 'fade-in-up 0.5s ease-out',
        'fade-out-up': 'fade-out-up 0.5s ease-out'
      },
      colors: { //custom colour palette for SK.
        'sk-primary-red': '#c66262',
        'sk-secondary-red': '#d14744',
        'sk-tertiary-red': '#fbefef',
        'sk-primary-blue': '#033076',
        'sk-secondary-blue': '#03307620',
        'sk-tertiary-blue': '#00537965',
        'sk-tertiary-blue': '#00537950',
        'sk-button-unselected': '#fafafa',
        'sk-button-selected': '#e5e5e5',
        'sk-navbar-text': '#777',
        'sk-navbar-text-hover': '#333',
        'sk-navbar-text-active': '#333',
        'sk-navbar-text-focus': '#333',
        'sk-divider-border': '#F0F0F1'
      }
    },
  },
  variants: {
    extend: {
      backgroundColor: ['active', 'disabled'],
      fontWeight: ['hover', 'focus'],
      textColor: ['disabled'],
      cursor: ['hover'],
      width: ['last']
    }
  },
  plugins: [],
}
```

```

return (
  <form onSubmit={submit}>
    <div className="bg-sk-tertiary-red m-auto self-center rounded pb-10 mx-20">
      <div>
        <p className="text-black text-center text-2xl font-bold mt-2py-2 p">>. Gaver og økonomiske
          bidrag</p>
      </div>

      {/Main wrapper/}
      <div className="flex flex-row m-auto w-2/3">
        <div
          className="mt-4 h-2/3 self-center justify-center bg-sk-button-selected border-2 border-sk-secondary-blue rounded-md shadow-2xl">
        </div>
      </div>
    </div>
  </form>
)

```

## Appendix 10 - Azure DevOps



Bachelorprosjekt Sanitetskvinnene Team

March 15 - March 26  
10 work days

Taskboard Backlog Capacity Analytics + New Work Item Column Options

Sprint 5 Person: All

Collaps all

New	Active	Resolved	Review	Closed
<p>241 Som system ønsker jeg å få mer coverage i forening</p> <p>Ejmar Risdalen</p> <p>State Closed</p>				<p>242 Frontend pipeline - Side opp side design og automatic routing of facts</p> <p>Ejmar Risdalen</p> <p>State Closed</p>
<p>246 Som admin ønsker jeg å smelt regioner på handlinger jeg gjør i grunnsett slik at jeg kan få bekreftelse på handlingene som er gjort</p> <p>Tym Stauheim</p> <p>State Closed</p>				<p>249 Somend pipeline - sette opp code coverage og automatisk testing</p> <p>Ejmar Risdalen</p> <p>State Closed</p>
<p>188 Som medlem av Kvinners sanitetsforening ønsker jeg å se informasjon tilknyttet min lokalforening slik at jeg kan melde eventuelle besvælinger til hovedkontoret</p> <p>Tym Stauheim</p> <p>State Closed</p>				<p>249 Visuell feedback lege kløring</p> <p>Tym Stauheim</p> <p>State Closed</p>
				<p>189 Lære og gå videre</p> <p>Ejmar Risdalen</p> <p>State Closed</p>

- Som en bruker av Kvinners Sanitetsforening ønsker jeg å kunne registrere eiendommer og virksomheter tilknyttet min forening slik at dette kommer med i årsrapporten
- Som en bruker av Kvinners Sanitetsforening ønsker jeg å kunne registrere hvilke gaver eller bidrag min forening har mottatt dette året slik at dette kommer med i årsrapporten
- Som system ønsker jeg å motta spørringer og levere informasjon tilknyttet forening- og medlemsinformasjon
- Som system ønsker jeg å motta spørringer og returnere informasjon tilknyttet gaver og økonomiske bidrag
- Oppdatere API kall til å sende med admin token
- Adminpanel tester backend

### Pipelines

Recent All Runs

All pipeline runs

Description	Stages	Full Request	Status
Adminpanel now checks for duplicate foreningsnummer and navn properly #202105112 on Sanitetskvinnene Backend Sanitetskvinnene Backend main 94c925c	✓	1199	✓ succeeded
Adminpanel now checks for duplicate foreningsnummer and navn properly #202105111 on Sanitetskvinnene Backend Sanitetskvinnene Backend main 94c925c	✗	1198	✗ failed
Removed rpm run testcd and test coverage steps until we #202105102 on Sanitetskvinnene Frontend Sanitetskvinnene Frontend cypress-pipeline 2128879	✓	1196	✓ succeeded
SK#381 Added test for aktivitet, gaverbidrag, koi, rapport, regnskapsstabell, adminpanel. #202105102 on Sanitetskvinnene Frontend Sanitetskvinnene Frontend main 8597c64	✗	1181	✗ failed
SK#359 changed from radio buttons to checkbox #202105101 on Sanitetskvinnene Frontend Sanitetskvinnene Frontend main 984c77b	✓	1176	✗ failed
SK#373 Created weblayer and controller tests for both pdf and excel generation #202105102 on Sanitetskvinnene Backend Sanitetskvinnene Backend main 544a7a9	✓	1171	✗ failed
varchar issue with annet. #202105101 on Sanitetskvinnene Backend Sanitetskvinnene Backend main 7224c11	✓	1164	✓ succeeded
SK#362 No longer a div the covers the content, so the footer is not 2 buttons #202102073 on Sanitetskvinnene Frontend Sanitetskvinnene Frontend main 4786c88	✓	1153	✓ succeeded

### Sanitetskvinnene Backend

Files

Contents History

Graph Commit

Commit	Full Request	Status
Adminpanel now checks for duplicate foreningsnummer and navn properly 94c925c · Einar W. Rødalen · Mon at 5:51 PM	1199	✓ succeeded
SK#373 Created weblayer and controller tests for both pdf and excel generation 544a7a9 · Eirikalen · Sat at 11:58 PM	1198	✓ succeeded
varchar issue with annet. 7224c11 · Eirikalen · Sat at 8:43 PM	1196	✓ succeeded
Alt og oppdaterte og createMtl was missing from Aktivitet migration file, also fixed test that got broken by last change. 7a2799d · Einar W. Rødalen · May 5 at 1:47 PM	1181	✓ succeeded
SK#368 PDF and Excel generation is now ready. /rapport/hent/<foreningsnummer> and /rapport/admin/ansrapport are the endpoints. 8874767a · Eirikalen · May 2 at 7:20 PM	1176	✗ failed
SK#388 Created rapport endpoint with controller and service, that generates and returns a PDF over HTTP. 6d7887c · Einar W. Rødalen · Apr 30 at 4:15 PM	1171	✗ failed
SK#321 Backend for Regnskap with tests completed. 889394d · Eirikalen · Apr 28 at 1:26 PM	1164	✓ succeeded
Added annetIdNet to the domain and dto object, as well as to the migration file. 7a2799d · Einar W. Rødalen · Apr 27 at 12:40 PM	1153	✓ succeeded
SK357 DTO ready 94c925c · Einar W. Rødalen · Apr 25 at 8:40 PM	1151	✓ succeeded
Saw that AktivitetController and LokalforeningController didnt really check the existence of a forening with the dto's foreningsNummer, so I changed that and added Repo.bils.foreningsNummer/NoFound! 93462a5 · Einar W. Rødalen · Apr 25 at 8:40 PM	1152	✓ succeeded

### Bachelorprosjekt Sanitetskvinnene

#### Sprint Retrospective 15.02

Trym Skarheim Feb 15

**Good:**

- Reestimerte backlog og fikk bedre oversikt.
- Godt miljø
- Kommet godt i gang.
- god mappe- og kodestruktur.
- gode rutiner.
- fin commit history
- daily standups
- avtalt tascotvedt

**Bad:**

- Dårlig estimering innledningsvis.
- Dårlig til å skrive timer
- Dårlig til å dokumentere
- Litt lite produktive.
- sette av mer tid til retrospekt
- sette av mer tid til demo.

**Can be improved:**

- Bli flinkere til å opprette mindre tasks.
- Bli flinkere til å føre opp timer
- Bli til å dokumentere underveis
- Bli mer produktive
- daily standups kan bli mer spesifikk.
- BLI ENIGE OM STANDUP RUTINE: mangel på struktur når standup.

3 visits in last 30 days

Add a comment...

## Appendix 11 - Test examples

Frontend:

```
describe("should not login", () => {
  it("should display wrong foreningsNummer or password", () => {
    cy.visit('http://localhost:3000')
    cy.wait(2500)
    cy.get('#username').type( text: "93457")
    cy.get('#password').type( text: "1231")
    cy.contains('Logg inn').should("be.visible").click()
    cy.get('#password-helper-text').should("be.visible")
    cy.contains( "Feil foreningsnummer eller passord").should("be.visible")
  })
})
```

Backend:

```
47 @BeforeEach
48 public void init() {
49     mockServer = MockRestServiceServer.createServer(testRestTemplate);
50     GaveRowDto row1 = new GaveRowDto( beskrivelse: "Ønsker et opplegg for gullkonfirmanter.", alternativ: "Eldre", beløp: 25000);
51     GaveRowDto row2 = new GaveRowDto( beskrivelse: "Julegave for mor som har alt allerede", alternativ: "nnet", beløp: 1000);
52     GaveRowDto row3 = new GaveRowDto( beskrivelse: "Penger for ribbe til det lokale food shelter", alternativ: "Krisesenter", beløp: 5000);
53     GaveRowDto[] rows = new GaveRowDto[]{row1, row2, row3};
54     valid = new GaveDto( foreningsNummer: 702044, rows, (row1.getBeløp() + row2.getBeløp() + row3.getBeløp()));
55     invalid = new GaveDto();
56 }
57
58 @Test
59 @FlywayTest
60 public void saveValidGave_returnNewGaver_And_OK() throws URISyntaxException, JsonProcessingException {
61     String endpoint = "http://localhost:8080/gaver/Lagre";
62     HttpHeaders headers = new HttpHeaders();
63     headers.setContentType(MediaType.APPLICATION_JSON);
64
65     HttpEntity<GaveDto> request = new HttpEntity<>(valid, headers);
66     mockServer.expect(ExpectedCount.once(), requestTo(new URI(endpoint)))
67         .andExpect(method(HttpMethod.POST))
68         .andRespond(withStatus(HttpStatus.OK)
69             .contentType(MediaType.APPLICATION_JSON)
70             .body(objectMapper.writeValueAsString(valid)));
71
72     ResponseEntity<GaveDto> actual =
73         testRestTemplate.postForEntity(endpoint, request, GaveDto.class);
74
75     mockServer.verify();
76     assertThat(actual.getBody()).isEqualTo(valid);
77     assertThat(actual.getStatusCode()).isEqualTo(HttpStatus.OK);
78 }
```